NR INDEKSU 353965
PL ISSN 0860-1674

7

Z MIKROKOMPUTEREM NA TY

Bajtek

MIESIĘCZNY DODATEK DO SZTANDARU MŁODYCH NR7(19) LIPIEC 1987 CENA 100 ZŁ



SM SZTANDAR MŁODYCH

EFEKTYWNOŚĆ I ELEGANCJA

ZAWODOWCY

WARSAW BASIC

DAN DARE

SPORTOWCY Z KOMPUTERA

Dobrze resorowany autobus, stoliki i krzesła zamiast ławek, na stolikach komputery osobiste, z przodu autokaru tablica... Załogę klasy na kółkach stanowi kierowca-elektromechanik (trzeba przecież podłączyć klasę do wiejskiej sieci elektrycznej lub uruchomić przenośny agregat) i instruktor-informatyk... Co stoi na przeszkodzie aby po polskich wsiach zaczęły jeździć takie komputerowe klasy na kółkach? — pytałem w tym miejscu dwa miesiące temu. I oto z satysfakcją stwierdzam, że nasza propozycja szybko znalazła społeczny odzew.

Otrzymaliśmy listy od nauczycieli i uczniów kilku szkół wiejskich, którzy gotowi są podjąć tą metodą naukę informatyki. W jednym przypadku otrzymaliśmy nawet plan szkolnego podwórka z zaznaczeniem, w którym miejscu można by nasz komputerowy autokar ustawić i jak podłączyć go do szkolnej sieci elektrycznej!

Najważniejsze jednak, że zgłosił się sponsor deklarujący gotowość zakupu i wyposażenia w sprzęt tego pierwszego komputerowego autobusu. Oto już po ukazaniu się tekstu przyszli do naszej redakcji koledzy Zbigniew Niemczynowicz i Aleksy Misiejuk reprezentujący Zespoły Usługowo--Wytwórcze Zarządu Krajowego ZMW „Agrotechnika". W naszym pomyśle „zakochali się od pierwszego wejrzenia". W imieniu swojej firmy zadeklarowali natychmiastowe wyłożenie środków finansowych i szybkie podjęcie prac organizacyjnych mających doprowadzić do szybkiego wyjechania w Polskę komputerowej „klasy na kółkach". Dziękujemy za błyskawiczną reakcję!

Warto przy okazji przypomnieć, że nie pierwsza to już tak cenna społecznie inicjatywa „Agrotechniki" w zakresie upowszechniania informatyki. Niedawno informowaliśmy o ufundowaniu przez nią i przekazaniu do kół ZMW aż 50 komputerów osobistych. Tak trzymać, koledzy! Czekamy teraz na zgłoszenia kolejnych sponsorów (jeden autobus to zaledwie kropla w morzu potrzeb polskiej wsi) i na... kolejne zgłoszenia szkół wiejskich. Musimy przecież opracować optymalny przebieg naszej autokarowo-komputerowej linii. W wakacje już pewnie nie zdążymy jej uruchomić, ale 1 września jest datą całkiem realną!

A'propos wakacji i komputerów. Próbowałem kupić niedawno w „Polresie" bilet z miejscówką, aby móc spokojnie wyjechać na kilka dni urlopu. I zalała mnie krew. Nie chodzi wcale o 3-godzinne czekanie w tropikalnym zaduchu źle wietrzonej sali na 1 piętrze Dworca Centralnego, naszego cuda techniki z drugiej połowy lat siedemdziesiątych. Chodzi o to, że panienki z okienka w dalszym ciągu, abym mógł dostać rezerwację muszą łączyć się telefonicznie z jakąś panienką, która ma prawdopodobnie na swym biurku listę wolnych miejscówek. Czasami próba dodzwonienia się do tego „centralnego banku danych" trwała i 40 minut!

Rezerwacja biletów jest najbardziej klasycznym ze wszystkich klasycznych miejsc, w których komputeryzację należy stosować w pierwszej kolejności. Przykład ten powinien się znaleźć już w podręcznikach dla pierwszej klasy szkoły podstawowej, gdyż jego logika jest tak oczywista, że nawet 7-latek zrozumie ją od razu. Dorosłym, jak się okazuje, zrozumieć to trudniej.

Jak dotychczas komputerowy system rezerwacji miejsc wprowadził tylko LOT, gdyż inaczej automatycznie pozostałby na uboczu światowego systemu komunikacji lotniczej. Mam nadzieję, że wobec takiej konieczności staną wkrótce i inni przewoźnicy, gdyż... szkoda gadać!

I jeszcze jedna wakacyjno-komputerowa uwaga. Znowu, jak w ubiegłym roku, zorganizowane zostanie w Polsce kilkaset obozów komputerowych. Bardzo się z tego cieszymy, gdyż skoro trudno jest spotkać komputer w szkole, to dobrze chociaż, że można to zrobić w czasie wakacji. Ambicją każdego większego zakładu przemysłowego jest zorganizowanie przynajmniej jednego obozu czy też kolonii „komputerowej". I bardzo dobrze. Jesteśmy ostatnimi, którzy mieliby cokolwiek przeciwko takim ambicjom. Tylko apelujemy — do uczestników i wychowawców — niech komputer nie przesłoni Wam widoku z wakacyjnego namiotu!

Wakacje to przede wszystkim odpoczynek, zabawa, sport. I komputery, oczywiście, ale obok, a nie zamiast zabawy na świeżym powietrzu. W innym przypadku wychowamy pokolenie komputerowych cherlaków. Rozwój intelektualny nie może odbywać się kosztem rozwoju fizycznego. Obie sfery powinny rozwijać się harmonijnie. Pamiętajcie o tym, drodzy wychowawcy.

Nasze wakacyjne hasło: komputerowcy najlepszymi sportowcami! Do biegu! Gotowi! Start!

Waldemar Siwiński



WYBIERZ SAM

„BAJTEK" — MIESIĘCZNY DODATEK DO „SZTANDARU MŁODYCH"
ADRES: 00-687 Warszawa, ul. Wspólna 61. Tel. 21-12-05
Przewodniczący Rady Redakcyjnej: Jerzy Domański--redaktor naczelny „Sztandaru Młodych".
ZESPÓŁ REDAKCYJNY: Waldemar Siwiński (z-ca redaktora naczelnego „SM" — kierownik zespołu „Bajtka"), Roman Poznański (z-ca sekretarza redakcji „SM" — sekretarz zespołu „Bajtka"), Krzysztof Czernek, Sławomir Gajda (red. techniczny), Andrzej Gogolewski, Andrzej Kowalewski, Andrzej Podulka, Sławomir Polak, Wanda Roszkowska (opr. graficzne), Kazimierz Treger, Marcin Waligórski, Roman Wojciechowski. Zdjęcia w numerze: Leopold Dzikowski.
Klany redagują:
Commodore — Klaudiusz Dybowski,
Amstrad-Schneider — Tomasz Pyć, Sergiusz Wolicki,
Spectrum — Konrad Fedyna, Michał Szuniewicz,
Atari — Wiesław Migut, Wojciech Zientara.

Fotoskład — Tadeusz Olczak,
Montaż offsetowy — Grażyna Ostaszewska,
Korekta — Maria Krajewska, Ewa Mowińska.
WYDAWCA: RSW „Prasa-Książka-Ruch" Młodzieżowa Agencja Wydawnicza, al. Stanów Zjednoczonych 53, 04-028 Warszawa. Telefony: Centrala 13-20-40 do 49, Redakcja Reklamy 13-20-40 do 49 w. 403, 414.
Cena 100 zł.
Skład techniką CRT-200, przygotowalnia offsetowa i druk: PRASOWE ZAKŁADY GRAFICZNE RSW „PRASA--KSIĄŻKA-RUCH" w Ciechanowie, ul. Sienkiewicza 51.
Żam. nr 090177, nakład 250 000 egz., K-109

SM SZTANDAR MŁODYCH

Bajtek



ROBOTNICZA SPÓŁDZIELNIA WYDAWNICZA
»PRASA • KSIĄŻKA • RUCH«

PRZYZNAJE NAGRODĘ
I STOPNIA

ZESPOŁOWI REDAKCYJNEMU
MIESIĘCZNIKA „BAJTEK"

ZA POPULARYZACJĘ INFORMATYKI
WŚRÓD DZIECI I MŁODZIEŻY

Prezes



Wiesław Rydygier

WARSZAWA, MAJ 1987 ROK

Uśmiechnięte twarze twórców „BAJTKA" oglądających siedemnasty już numer miesięcznika mają związek z osiągnięciem niebagatelnego sukcesu. Zespół redakcyjny uhonorowany został Nagrodą Prezesa RSW „Prasa Książka Ruch" I Stopnia za popularyzację informatyki wśród dzieci i młodzieży.

KTO KUPUJE STUDENCKIE PROGRAMY?



Rozmowa z JERZYM DOMAŃSKIM, dyrektorem Warszawskiego Centrum Studenckiego Ruchu Naukowego.

— Czytelnicy „Bajtka", to, na ogół licealiści, bądź uczniowie szkół podstawowych, a studenci? Czytają „Komputer", czy też w ogóle nie interesują się informatyką?

— Na giełdach obok handlarzy dominują rzeczywiście 14-16-latki. Czy oznacza to brak zainteresowania studentów informatyką? W pewnym stopniu tak. Entuzjazmu dla niej na uczelniach raczej nie widać. Dla przykładu na Politechnice Warszawskiej wyróżnić można w zasadzie dwie grupy. Jedna, to studenci wydziałów „nieelektronicznych". Większość z nich umie napisać prosty program w BASIC-u i niewiele ponadto. Natomiast druga grupa, wśród której dominują studenci elektroniki, to w zasadzie profesjonaliści. Już na studiach potrafią oni sprzedawać swoje umiejętności programowania i to za duże pieniądze.

— Czy właśnie brak powszechnego „entuzjazmu komputerowego" był powodem tak późnego zajęcia się informatyką w Centrum?

— Dopiero przed rokiem, gdy staliśmy się firmą ZSP na własnym rozrachunku uzyskaliśmy potencjalną możliwość zdobycia sprzętu. W trzy miesiące później ruszyły już kursy programowania w wyposażonej przez nas sali. Dziś pracownia komputerowa zapewnia 50 proc. obrotu całej firmy. Prowadzimy wspomniane kursy BASICA, PASCALA, C, Assemblera na komputerach IBM i AMSTRAD a także kursy obsługi IBM-ów. Przede wszystkim jednak na zlecenie różnych zakładów pracy organizujemy im systemy zarówno od strony sprzętowej jak i przede wszystkim softwareu.

— Znaleźliście zatem chętnych do jego opracowania?

— Mieliśmy trochę szczęścia. Ludzie, którzy u nas pracują nie mogą wprawdzie zarobić tyle, co w innych firmach — obowiązuje nas stawka 230 zł za godzinę pracy programisty — lecz widocznie podoba im się panująca tu fajna atmosfera, możliwość samodzielnego znajdowania i wykonywania atrakcyjnych a jednocześnie pożytecznych prac.

— Jakie są ich efekty?

— Myślę, że nie najgorsze, porównując choćby z wynikami innych producentów oprogramowania. Na targach „Intersoft" we Wrocławiu nasze pakiety były w każdej chwili do zaprezentowania na stoisku. W odróżnieniu od programów wielu innych znacznie głośniejszych firm, które po prostu „nie chodzą". W warszawskich Zakładach „Syrena" musieliśmy dla przykładu niemal od początku przerabiać sprzedany im za naszym pośrednictwem pakiet finansowo-księgowy produkcji najgłośniejszej aktualnie w Polsce firmy softwarowej. Teraz działa już bez zarzutu. Nauczeni jednak doświadczeniem sprzedajemy obecnie wyłącznie własne oprogramowanie.

— I jest na nie popyt?

— Oczywiście. Powoli wszyscy zaczynają się już orientować, że komputera bez oprogramowania kupować nie należy. Często nasi kontrahenci zamawiają najpierw programy, a dopiero przy ich odbiorze każą sobie dodać do nich... komputer. Takie transakcje wiązane są zresztą dla nas najbardziej opłacalne — programy są wciąż jeszcze na naszym

rynku nieproporcjonalnie tanie w stosunku do ceny sprzętu i wyłącznie na nich trudno zarobić.

— Kto kupuje studenckie programy?

— Wszyscy. Najciekawszą współpracę nawiązaliśmy jednak ze... spółdzielniami rolniczymi tworzącymi spółkę „Agrosoft". Opracowujemy dla niej systemy: finansowo-księgowy, ewidencji materiałowej i płac. Tworzenie ich nie jest rzeczą łatwą, bowiem w spółdzielczości rolniczej obowiązują inne niż w pozostałych jednostkach gospodarczych system ekonomiczno-finansowy. Pracujemy także nad programami ściśle rolniczymi.

— Proponujecie komputer zamiast traktora?

— Nie można powiedzieć z góry, co jest ważniejsze. W spółdzielni powinno być i jedno i drugie. Gdy trzeba wybierać wówczas decydować musi rachunek spodziewanych zysków. Nierzadko przemawiać on będzie na korzyść komputera. Nasi potencjalni klienci też są o tym przekonani. Mieliśmy już zresztą możliwość zaprezentowania im pierwszych wersji przyszłych programów na wspólnym spotkaniu w Karpaczu pod hasłem „narty i komputery".

— Narty miały być premią dla programistów?

— Tylko takie możemy im zaproponować. Wspomniałem już, że ci spośród studentów i młodych pracowników nauki, którzy są naprawdę dobrzy i wiedzą na co ich stać mogą w innych miejscach zarobić wielokrotnie więcej.

— Czyżby studia stały się wreszcie opłacalną inwestycją?

— Na takich wydziałach jak elektronika informatyka czy matematyka stosowana niewątpliwie tak, przynajmniej w Warszawie. Najlepsi bez żadnych kombinacji zarobić mogą i 100 tys. zł miesięcznie. Ale tych naprawdę dobrych nie jest jeszcze zbyt dużo. Może będzie ich więcej, gdy na studia trafią czytelnicy „Bajtka". Tutaj, w Centrum zawsze mogą liczyć na to, że znajdą warunki do podniesienia swoich umiejętności.

— Zgoda, że to, jak przygotowani będą do pracy z komputerem młodzi ludzie trafiający na studia, zależy także od „Bajtka". Ale i wy moglibyście dołożyć swoją cegiełkę".

— Prowadzone u nas kursy programowania są otwarte także dla uczniów szkół średnich. Poza tym myślimy poważnie o rozpoczęciu produkcji programów edukacyjnych. Kłopot w tym, że nie wiadomo jeszcze na jaki komputer. Przed rokiem z inicjatywy Rady Okręgowej ZSP w Warszawie spotkaliśmy się z dyrektorem Zrzeszenia „Mera". Na tym spotkaniu ustaliliśmy, że rozpoczniemy prace nad programami dla szkół. W ciągu dwóch, trzech tygodni otrzymać mieliśmy ze Zrzeszenia pierwsze dwa „Elwro 800 Junior" I co? Minął rok, wciąż jesteśmy w kontakcie, ale... Zrzeszenie samo jeszcze nie dysponuje nawet kilkoma egzemplarzami „Juniorów". Skompletowaliśmy zatem zespół, ułożyliśmy nawet scenariusz pracy i... czekamy.

— Tylko czekacie?

— W tej sprawie sami wiele zrobić nie możemy. Myślę jednak, że połączonymi siłami różnych niewielkich firm takich jak nasza, zatrudniających niemal samych młodych ludzi, przełamać będzie można i w tej sprawie barierę niemożności. Zamierzamy zarejestrować nasze Centrum jako jednostkę innowacyjną natychmiast po tym, gdy tylko Sejm przyjmie odpowiednią ustawę. Jesteśmy członkami powstałej niedawno Polskiej Izby Jednostek Innowacyjnych. Jeśli okaże się, że „komputer dla każdego" wciąż pozostaje poza zainteresowaniem wielkich zakładów elektronicznych być może właśnie jednostki innowacyjne przyczynią się do uruchomienia jego produkcji. Tak, czy inaczej z komputerami już się nie rozstaniemy.

Rozmawiał:
Grzegorz Onichimowski

SWEGO NIE ZNACIE

WŁAMANIA NIE BĘDZIE

Kilka lat temu grupa amerykańskich informatyków postanowiła udowodnić, że opieranie bankowości wyłącznie na komputerach może skończyć się katastrofą. Swój dowód przeprowadzili w najprostszy sposób — włamali się nocą do sieci komputerowej i zmienili zaprogramowany kurs dolara. Oczywiście bankierzy o całej operacji wiedzieli, bo gdyby podobnego wyczynu dokonał informatyk-przestępca nazajutrz dolar byłby tańszy od śmieci. W naszym kraju takie włamanie jest absolutnie niemożliwe. Niestety, nie świadczy to o wyższości naszych banków. Po prostu nie istnieje sieć, do której można by się włamać.

O pożytkach płynących z zastosowania techniki komputerowej w bankach większa część świata przekonała się już dość dawno. Dziś sprawą zupełnie normalną jest to, że dżentelmen posiadający konto na Wall Street pobiera (wpłaca, przelewa, odpisuje) pieniążki np. na Alasce.

A u nas? Wybrałem się do jednego z warszawskich banków, o którym mówiono mi, że „ma komputer". Faktycznie „ma". I to wszystko co można powiedzieć o skomputeryzowaniu tego oddziału. Pomijając fakt, że poczciwa MERA 9150 (tak, tak) akuratnie była zepsuta, jej pomoc jest prawie żadna. Komputer spełnia tu rolę mechanicznego archiwisty, statystyka i głównego księgowego. Zapisuje on wszystkie przeprowadzone w oddziale operacje, które jednak wcześniej muszą być przygotowane przez pracownika „na piechotę".

Jedyną zaletą jest zmniejszenie liczby pomyłek, bo komputer nie da się łatwo oszukać i nie przyjmie np. numeru nie istniejącego konta bankowego. No i obsługa jest przyjemniejsza niż zwykłego liczydła czy tabulatora — miękka klawiatura, ekran, który czasem zapiszczy, sympatycznie zamruga do kasjerki...

Na szczęście są w Warszawie banki, gdzie komputer wykonuje działania prawie takie same, jak jego „koledzy" z innych zakątków świata. Takie banki są dwa — przy ul. Sienkiewicza i Rotunda. Odwiedziłem ten drugi.

W październiku 1979 roku Rotunda zaczęła na nowo funkcjonować po tragicznej eksplozji. I rzeczywiście oznaczało to dla tego oddziału „nowe". Wówczas po raz pierwszy w naszym kraju do banku zawitał komputer. Zresztą w tej dziedzinie Rotunda do dziś jest oddziałem wzorcowym. Pracuje tu amerykański system MST, czyli Modulowany System Terminalowy, oparty na komputerach NCR-2500. Zdanie pracowników, którzy nie wyobrażają sobie pracy z liczydłem i stemplem w ręku (jak dawniej), świadczy, że jest on pożyteczny.

Konkretnie? System znakomicie zdaje egzamin przy operacjach dopisywania odsetek, dużych wypłatach z książeczek obiegowych oraz wymiany tych książeczek obiegowych. W normalnym banku zabieg taki trwa około dwóch tygodni, podczas gdy tutaj — pięć minut.

Jeżeli książeczka jest założona w Warszawie, albo najbliższej okolicy, wystarczy „wczytać" dane klienta, a komputer błyskawicznie informuje, czy taka książeczka istnieje, podaje stan wkładu oraz dokonuje operacji wpłaty lub wypłaty określonej sumy. Trwa to dwie-trzy sekundy. Zapis wykonywanej operacji kasjer otrzymuje na małej drukarce, podczas kiedy inna wykonuje zapis w książeczce. Sprawa komplikuje się, gdy mamy do czynienia z książeczką założoną — powiedzmy — w Szczecinie. Przypomina to komputerowe kasy biletowe, gdzie otrzymujemy ślicznie wydrukowany przez komputer bilet, ale pani wcześniej musi telefonicznie uzgodnić czy dane miejsce nie jest zajęte. Tutaj kasjerka także chwyta za telefon i dzwoni do Szczecina, aby potwierdzić stan wkładów. Nie dotyczy to sum bieżących do 50 tys. złotych.

Kolejnym przykładem działania komputera-bankiera jest prowadzenie kontroli tabel premiowych bonów oszczędnościowych. Komputer ma zaprogramowaną treść wszystkich tabel i wprowadzenie symbolu danego bonu daje odpowiedź czy został on wylosowany, a jeżeli tak, to na jaką sumę. Od chwili zatrudnienia przy tej czynności komputera nie zdarzyła się jeszcze ani jedna pomyłka z wypłacaniem premii.

Ogromnym uproszczeniem jest zwolnienie kasjerki z obowiązku prowadzenia całej buchalterii biurowej. Nie musi ona na osobnych kontrolkach notować wpłat za energię elektryczną, wodę, gaz, psa i telewizor... Komputer wszystkie operacje notuje na tzw. kontrolce zbiorczej i dostarcza w postaci wydruku. Po zakończeniu pracy kasjerka musi jedynie policzyć pieniądze i uzgodnić posiadaną gotówkę z saldem, jakie powinna mieć na koniec dnia.

Aby pracować w Rotundzie przy klawiaturze komputera należy przejść specjalne miesięczne przeszkolenie i zdać wewnętrzny egzamin. Taka specjalizacja wystarczy. Natomiast konserwacją, programowaniem i naprawą zajmują się specjaliści z Centrum Elektronicznego NBP. Choć awaryjność systemu jest naprawdę nieznaczna i nie osiąga nawet jednego procenta w stosunku do obsługiwanych każdego dnia pięciu tysięcy klientów.

Bankierzy z Rotundy mają więc pracę, która dzięki komputerowi wcale nie musi być nudnym „urzędoleniem", choć pewnie mają kolejne życzenie — stworzenie sieci komputerowej i przeprowadzanie na tym urządzeniu wszystkich operacji. Pocieszeniem może być fakt, że inni dopiero od nich się uczą. Na przykład kasjerzy z X OM NBP PKO przy Placu Powstańców. Właśnie ten oddział stanie się pod koniec roku trzecim BIT-bankiem. Może jest to krok na drodze pełnej komputeryzacji polskiej bankowości? Odpowiedzieć mogą specjaliści z Departamentu Informatyki NBP.

Janusz Janiec

Turbo PASCAL został, w stosunku do standardowej wersji PASCAL-a, wzbogacony o nowe elementy, których zadaniem jest zapewnienie użytkownikowi możliwości tworzenia nowoczesnego oprogramowania.

TURBO PASCAL
I COŚ JESZCZE

Zmiany i rozszerzenia języka w Turbo PASCAL-u

KOMENTARZE

Podobnie jak w wielu innych nowoczesnych kompilatorach **PASCAL-a,** komentarze programu mogą służyć do przekazywania dodatkowych poleceń dotyczących samej kompilacji. Komentarz taki powinien rozpoczynać się od znaku $, po którym następuje (nie poprzedzone odstępem) jednoliterowe oznaczenie polecenia, i ewentualnie parametr, którym najczęściej jest znak + lub –. Można umieścić w jednym komentarzu listę kilku takich dyrektyw oddzielonych przecinkami, np.

{$R–,X+,V–}

Bodaj najistotniejszą z dostępnych dyrektyw kompilacji jest możliwość kompilowania tekstu programu zawartego na różnych plikach przy pomocy dyrektywy {$I}. Jej parametrem jest nazwa pliku z tekstem źródłowym. W momencie napotkania powyższego polecenia kompilator wczytuje żądany plik z dysku, a następnie kompiluje go — tak, jakby był integralną częścią programu źródłowego. Umożliwia to oszczędną gospodarkę pamięcią komputera, tworzenie programów dłuższych niż objętość edytora i dodatkowo tworzenie np. bibliotek procedur. Oto przykład użycia dyrektywy I:

```
program Wykres;
  {$I Procedury.pas}
  begin
    {$I Grafik.pas}
  end.
```

Dyrektyw I nie można zagnieżdżać. Oznacza to, że dla danego programu wolno ich użyć tylko w jednym z jego plików źródłowych.

A oto inne podstawowe dyrektywy kompilacji w Turbo PASCAL-u:

- **{$I–}** — Odłączenie sygnalizacji błędów operacji wejścia/wyjścia. W przypadku odłączenia błędy takie są sygnalizowane tylko przy pomocy funkcji IOResult. Nie należy mylić niniejszej dyrektywy z opisaną wyżej dyrektywą łączenia plików.
- **{$R+}** — Włączenie sygnalizacji błędów przekroczenia zakresu indeksu w tablicy, przekroczenia zakresu dozwolonego przez typy zmiennych itp. Przy opcji wyłączonej program działa szybciej.
- **{$V–}** — Wyłączenie dokładnej kontroli zgodności typów zmiennych typu String w deklaracjach i wywołaniach procedur. Po wyłączeniu dozwolone jest użycie jako parametru typu String dowolnej zmiennej tego typu (bez względu na jej zadeklarowaną długość).
- **{$U+}** — umożliwienie użytkownikowi przerwania wykonania gotowego programu przy pomocy klawiszy CTRL + C. Program z włączoną opcją działa wolniej.
- **{$X–}** — Wyłączenie opcji szybkiego dostępu do tablic. Po wyłączeniu kompilator minimalizuje obszar pamięci zajmowany przez tablice, kosztem szybkości dostępu do ich elementów.

NAGŁÓWEK PROGRAMU

Może zostać pominięty.

ETYKIETY

Mogą mieć postać nie tylko liczb, jak w PASCAL-u standardowym, lecz również dowolnych identyfikatorów, np.

label BladWeWy, 999;

Etykiety są „widoczne" tylko w jednej jednostce programowej. Nie jest zatem możliwe wykonanie skoku **(goto)** np. na zewnątrz procedury lub funkcji.

STAŁE

Dostępne są dwie dodatkowe stałe standardowe:

- **Pi** — Przybliżenie liczby Pi, dokładniej 3.1415926536,
- **MaxInt** — Maksymalna dostępna liczba całkowita = 32767.

STAŁE OPATRZONE TYPEM

Stałe opatrzone określeniem typu występują wyłącznie w Turbo PASCAL-u. Ich użycie pozwala na pewne oszczędności pamięci i umożliwia użycie jako stałych złożonych struktur danych, np. tablic. Stałe standardowe są w kodzie wynikowym umieszczane w całości tyle razy, ile razy nastąpi do nich odwołanie; stałe opatrzone typem zajmują miejsce w pamięci tylko raz — w miejscu, w którym są definiowane.

Definicja takiej stałej ma postać podobną do poniższych:

```
const Tytul: string [8] = 'PRZEKRÓJ';
      g : real = 9.81;
      Break: char = 'Q';
```

Stałe opatrzone typem są traktowane w programie tak jak zmienne (z nadaną wartością początkową). Z tego powodu nie wolno ich używać w miejscach, w których wartość stałej musi być znana w czasie kompilacji, np.. w definicjach typów. Stałymi opatrzonymi typem mogą być również tablice, rekordy i zbiory. Ze względu na konieczność określania typów, blok definicji stałych z typami wolno umieścić po definicjach typów programu (procedury, funkcji). Na przykład:

```
type Stan = (Dziala, NieDziala, Zepsuty);
     Komunikaty = array [Stan] of string [11];
     Punkt = record
               X,Y : real;
             end;
const CoJest : Komunikaty =
               ('DZIALA','NIEDZIALA',
                'KONSERWACJA');
      Początek: Punkt = (X:0.0, Y:0.0);
    Samogloski : set of 'A'..'Z' =
               ['A','E','I','O','U','Y'];
```

TYPY DANYCH

W Turbo PASCAL-u wprowadzono nowe standardowe typy danych:

- **byte** — zdefiniowany jako 0..255,
- **string** — opisany osobno poniżej.

W odróżnieniu od standardowego PASCAL-a, Turbo pozwala na zamianę wartości w zasadzie dowolnego typu prostego na inny typ prosty (wyjątkiem jest tylko typ real). Dokonuje się tego, używając identyfikatora typu jako funkcji konwersji. Np. w odniesieniu do poniższych typów:

```
type Kierunek = (N, E, S, W);
     Wielkie  = 'A'..'Z';
```

zachodzą równości:

```
Integer (N)   = 0
Kierunek (3)  = W
Wielkie (15)  = '0'
Wielkie (N)   = 'A'
```

oraz np.

```
Char (55)     = '7'
Integer ('N') = 78
```

W celu konwersji na typ integer można zawsze użyć znanej już funkcji ord.

```
Ord (S)   = 2
Ord ('N') = 78
```

LICZBY CAŁKOWITE

Możliwe jest używanie liczb całkowitych w zapisie szesnastkowym. Zapis liczby należy w tym przypadku poprzedzić znakiem $, np.

Adres: = $1F00;

Dopuszczalnym zakresem liczb całkowitych w zapisie szesnastkowym jest $0000..$FFFF.

Oprócz zwykłych działań arytmetycznych Turbo PASCAL udostępnia pewne operacje na liczbach całkowitych, charakterystyczne dla języków niskiego poziomu.



a shl b — Przesunięcie bitowej reprezentacji liczby a o b bitów w lewo, np.

1 shl 7 = 128

a shr b — To samo z przesunięciem w prawo.

a and b — iloczyn logiczny wszystkich bitów reprezentacji liczb a i b, np.

12 and 22 = 4

a or b — Suma logiczna wszystkich bitów reprezentacji liczb a i b, np.

12 or 22 = 30

a xor b — Operacja alternatywy wykluczającej na bitach liczb a i b, np.

12 xor 22 = 26

Operacji **xor** wolno również, obok innych operacji logicznych, używać w odniesieniu do typu boolean:

true xor false = true

not a — Negacja wszystkich bitów reprezentacji liczby a, np.

not $0001 = $FFFE
not – 15 = 14

ZBIORY

Zbiory w Turbo mogą zawierać do 256 elementów. Zbiór mogą tworzyć wyłącznie elementy tego samego typu prostego.

ŁAŃCUCHY ZNAKÓW — TYP STRING

Typ standardowy string służy do reprezentacji tekstów. Jego wprowadzenie stanowi duże udogodnienie dla programisty, który w standardowym PASCAL-u musiał przechowywać ciągi znaków w tablicach. Deklarując użycie typu string należy podać maksymalną długość łańcucha znaków, jaki ma być reprezentowany, np.

```
type BuforTekstowy = string [128];
var Linia: string [80];
```

Podana długość maksymalna musi należeć do zakresu 1..255. Zmienne typu string mogą przyjmować wartość dowolnego łańcucha znaków o długości nie większej niż zadeklarowane maksimum. W przypadku próby przypisania dłuższego łańcucha jego koniec jest pomijany. Oto przykłady:

```
Linia: = '';
Linia: = 'Telefon 110';
read (Linia);
```

Dane typu string można porównywać przy pomocy operatorów =, <, <=, >, >=. Można też używać operacji + w celu łączenia łańcuchów ze sobą, np.

```
Linia: = 'rok 19' + '87:' + Linia;
```

Dozwolone jest podstawienie za wartość zmiennej typu string wyrażenia typu char lub array [..] of char. Wolno również odwoływać się do poszczególnych znaków łańcucha przy pomocy indeksów — jak w tablicy, np.

```
Znak: = Linia [10];
```

Pierwszy znak łańcucha nosi indeks 1. Pod indeksem 0 umieszczony jest znak odpowiadający bieżącej długości łańcucha.

Dalszych operacji na danych typu string można dokonywać przy pomocy standardowych procedur i funkcji:

Delete (Zmienna, OdMiejsca, IleZnakow)
Usuwa z łańcucha będącego wartością podanej zmiennej podaną liczbę znaków, począwszy od wskazanego miejsca.

Insert (Lancuch, Zmienna, OdMiejsca)
Wstawia podany łańcuch znaków do łańcucha będącego wartością podanej zmiennej, w miejscu wskazanym przez liczbę OdMiejsca.

Str (Liczba, Zmienna)
Konwersja wartości podanej liczby (typu **real** lub **integer**) na łańcuch znaków oraz nadanie zmiennej typu **String** wartości tego łańcucha.

Val (Lancuch, ZmiennaLiczbowa, Kod)
Konwersja zapisu liczby w postaci łańcucha na postać typu **integer** lub **real** i przypisanie wyniku zmiennej ZmiennaLiczbowa (typu **integer** lub **real**). Zmiennej całkowitej **Kod** przypisywane jest 0, jeżeli w trakcie konwersji nie wystąpił błąd. W przeciwnym przypadku zmienna ta otrzymuje wartość miejsca w łańcuchu, w którym błąd wystąpił.

Copy (Lancuch, OdMiejsca, IleZnakow)
Funkcja przyjmująca wartość fragmentu podanego łańcucha o podanym początku i długości.

Concat (Lancuch1, Lancuch2, ...)
Funkcja odpowiadająca operacji

Lancuch1 + Lancuch2 + ...
Ilosc argumentów jest dowolna.

Lenght (Lancuch)
Długość podanego łańcucha.

Pos (Wzorzec, Lancuch)
Funkcja przyjmująca wartość miejsca, w którym występuje podany wzorzec wewnątrz podanego łańcucha. Jeżeli wzorzec nie występuje, wartość funkcji wynosi 0.

PLIKI

Pliki definiuje się podobnie jak w PASCAL-u standardowym. Nie jest wymagane umieszczanie nazw używanych zmiennych plikowych w nagłówku programu. Dostępny jest typ standardowy **text**, określony jako **file of char**.

W Turbo występują (oprócz **Input** i **Output**) następujące standardowe zmienne plikowe:

Con — plik odpowiadający konsoli użytkownika,
Kbd — klawiatura tejże konsoli (wolno tylko czytać),
Lst — drukarka.

W Turbo PASCAL-u występuje tylko jeden rodzaj plików, mianowicie stałe pliki dyskowe. Zrezygnowano wobec tego z operacji **put** i **get**, występujących w standardzie PASCAL-a i odnoszących się głównie do plików tymczasowych. Ich rolę pełnią procedury **read** i **write**.

W związku z tak określonymi założeniami wprowadzono szereg dodatkowych procedur umożliwiających efektywne posługiwanie się plikami.

Assign (Zmienna, NazwaPliku)
Przypisanie danej zmiennej plikowej do pliku dyskowego o nazwie NazwaPliku, będącej łańcuchem znaków, np.

Assign (Spis, 'KARTOTEKA.TXT');
Przypisanie takie powinno być dokonane w programie przed pierwszym użyciem zmiennej plikowej **Spis** (np. **reset (Spis)** lub **rewrite (Spis)**).

Close (Zmienna)
Zamknięcie pliku określonego przez zmienną Zmienna. Zamknięcie musi być ostatnią operacją wykonywaną na każdym przetwarzanym w programie pliku.

Seek (Zmienna, N)
Ustawienie wskaźnika pliku Zmienna na N-ty element tego pliku. Pierwszemu elementowi pliku odpowiada N = 0.

Flush (Zmienna)
Wypisanie sektora pliku (bufora) aktualnie przetwarzanego w pamięci na dysk.

Erase (Zmienna)
Usunięcie z dysku pliku odpowiadającego zmiennej Zmienna.

Rename (Zmienna, NowaNazwa)
Zmiana nazwy pliku odpowiadającego zmiennej Zmienna na nową, określoną przez łańcuch NowaNazwa, np.

```
Identyfikator: = 'NOWY.TXT';
Rename (Spis, Identyfikator);
```

FilePos (Zmienna)
Funkcja przyjmująca wartość aktualnej pozycji wskaźnika pliku (patrz **Seek**).

FileSize (Zmienna)
Funkcja przyjmująca wartość liczby elementów danego pliku. Plikowi pustemu odpowiada wartość 0.

IOResult
Funkcja przyjmująca wartość kodu błędu ostatnio wykonanej operacji na pliku dyskowym (np. brak pliku o wskazanej nazwie, przepełnienie dysku, błędne podanie nazwy pliku). Jeżeli operacja odbyła się bezbłędnie, **IOResult** przyjmuje wartość 0. Funkcja ta umożliwia obsługę błędów tego rodzaju przy wyłączonej obsłudze automatycznej (dyrektywa **{$I–}**).

ZMIENNE STANDARDOWE

Oprócz wymienionych standardowych zmiennych plikowych Turbo zawiera tablicę standardową **Mem**. Dla pewnej wartości indeksu N **Mem [N]** oznacza zawartość komórki pamięci o adresie N. Umożliwia to użytkownikowi bezpośredni dostęp do pamięci, np.

```
Mem [$FFFA] : = $01;
Bajt : = Mem [32767];
```

PROCEDURY I FUNKCJE

W porównaniu ze standardem deklaracje procedur i funkcji pozostają w zasadzie nie zmienione. Nowością jest natomiast możliwość stosowania nakładek, tzn. generowania kodu wynikowego wybranych procedur na osobne pliki dyskowe, których zawartość w trakcie wykonywania programu byłaby w odpowiednim momencie ładowana i wykonywana. Rozwiązanie takie pozwala na oszczędną gospodarkę pamięcią w trakcie wykonywania programu i wykonywanie programów dłuższych niż mieści pamięć komputera.

Zamiar kompilowania procedury lub funkcji na osobny plik określamy poprzedzając nagłówek jej deklaracji słowem kluczowym **overlay**, np.

```
overlay procedure Edytor (var F : text);
```

Kompilator sam utworzy odpowiednią liczbę plików o unikalnych nazwach. Nakładki, w odróżnieniu od dyrektywy **{$I}** kompilatora, mogą być zagnieżdżane.

PROCEDURY I FUNKCJE STANDARDOWE

Oprócz wymienionych standardowych procedur i funkcji obsługi plików i zmiennych typu **string** Turbo PASCAL oferuje szereg procedur — w większości ułatwiających użycie ekranu monitora. Są to:

ClrEol
Skasowanie znaków w linii na prawo od kursora.

ClrScr
Skasowanie całego ekranu i umieszczenie kursora w jego lewym górnym rogu.

DelLine
Skasowanie linii w której stoi kursor i przesunięcie części ekranu poniżej o jeden wiersz w górę.

InsLine
Wstawienie pustej linii w miejscu kursora. Linie poniżej są przesuwane o jeden wiersz w dół.

GotoXY (Wers, Kolumna)
Przesunięcie kursora do wskazanego rzędu i kolumny ekranu. Górnemu lewemu rogowi odpowiadają współrzędne (1,1).

Delay (N)
Wstrzymanie wykonywania programu na ok. N milisekund.

Randomize
Inicjacja generatora liczb losowych.

InLine (Kod)
Procedura umożliwiająca łączenie PASCAL-a z kodem maszynowym. Parametrem **InLine** jest ciąg wartości odpowiadających programowi w kodzie maszynowym, przedzielanych znakiem **/**. W kodzie maszynowym wolno umieszczać nazwy zmiennych używanych na poziomie PASCAL-a. Wolno też wykorzystywać wszystkie rejestry procesora bez żadnych ujemnych skutków. Oto trywialny przykład wykorzystania InLine:

```
procedure Increment (var X : byte);
begin
    inline ($04/ X {INC X });
end;
```

Zbiór funkcji standardowych rozszerzono o nowe funkcje:

KeyPressed
Funkcja określająca, czy w danym momencie wciśnięty jest jakiś klawisz. Np.

```
repeat until KeyPressed;
read (Kbd, Klawisz);
```

Random
Rzeczywista liczba losowa z przedziału 0..1 (typem wyniku jest **real**).

Random (N)
Liczba naturalna wylosowana z przedziału 0..N-1. Typem wyniku jest **integer**).

Hi (Liczba)
Starszy bajt reprezentacji danej liczby całkowitej.

Lo (Liczba)
Młodszy bajt reprezentacji danej liczby całkowitej.

Swap (Liczba)
Liczba całkowita powstała przez zamianę starszego i młodszego bajtu argumentu.

Podany tu opis nie jest jeszcze kompletny. Poszczególne implementacje Turbo dają programiście możliwość np. operowania grafiką, złożonych operacji na pamięci, przetwarzania dowolnych plików dyskowych itp. Zainteresowanych odsyłam do podręcznika „Turbo Pascal Reference Manual", Borland International, Scotts Valley, California. Wszystkim natomiast życzę wiele satysfakcji z posługiwania się tym językiem — i wielu efektownych programów.

Marcin Waligórski

CeBit 87'

ATARI WYGRYWA PRZED STARTEM

W dniach od czwartego do jedenastego marca Hanower stał się po raz drugi miejscem spotkania międzynarodowych wystawców, którzy prezentowali nowinki z „komputerowego światka".

W porównaniu z rokiem 1986 liczba uczestników targów spoza RFN wzrosła z 375 do 760, powierzchnia wystawowa zajmowała 205 000 metrów kwadratowych, a zwiedzających było około 400 000.

W pokazie brała również udział jedna z największych firm komputerowych na świecie na czele z Jackiem, Samem i Leonardem Tramielem. Mowa oczywiście o Atari. Firma ta zamknęła rok 1986 wzrostem zysku o 81,8%. Pomijając Europę, gdzie komputery ATARI są jednymi z najlepiej sprzedających się, także w USA dzięki konkurencyjnym cenom jest to coraz bardziej popularny komputer. W Hanowerze pokazano nowe produkty korporacji: klon IBM PC i nową generację serii ST czyli MEGA ST.

ATARI PC jest kompatybilny ze standardem MS/DOS. Zegar procesora 8088-2 jest przełączalny między 8.0 a 4.77 MHz. Pozostawiono pustą podstawkę do dołączenia koprocesora 8087 przeznaczonego do obliczeń matematycznych. W podstawowej konfiguracji ATARI PC ma pamięć 512 KB RAM z możliwością rozszerzenia o 640 KB na płycie głównej. Komputer wyposażony jest seryjnie w sterownik graficzny, który emuluje standardy EGA, CGA i Hercules oraz umożliwia dostęp do dodatkowych 256 KB VIDEO-RAM. ATARI PC posiada wbudowaną stację dysków 5 1/4 cala o pojemności 360 KB oraz wejścia: równoległe, szeregowe i zintegrowane (dla myszy). Rozdzielczość w trybie tekstowym wynosi 85 znaków w 25 liniach, a w trybie graficznym:

a) z monitorem monochromatycznym 720×348 (HERCULES)

b) monitor kolorowy umożliwia wyprowadzenie 640×350 punktów w 16 kolorach z palety 64.

Istnieje możliwość podłączenia urządzeń peryferyjnych takich jak: 3,5 calowa stacja dysków, mysz i dysk twardy (Winchester). Ciekawostką może być fakt, że mysz do ATARI PC jest zgodna ze standardem ATARI. Komplet składa się z monochromatycznego monitora, oddzielnej klawiatury i komputera. Koszt w RFN około 1798 marek.

Drugą nowinką, zaprezentowaną przez firmę ATARI w Hanowerze było MEGA ST. Komputer pokazany w trzech wersjach: rozszerzenie pamięci do 1 MB, 2 MB, 4 MB RAM. Ta nowa generacja serii ST posiada niezależną klawiaturę połączoną z systemem spiralnie skręconym kablem. Nowa klawiatura jest bardzo funkcjonalna i wygodna w użytkowaniu. Jednostka centralna oparta na procesorze MOTOROLA 68 000 ma wbudowaną dwustronną stację dyskietek 3,5 cala o pojemności 720 KB. Pamięć przy MEGA ST 2 i 4 wyposażona w jeden Mega Bit-Chip. Płyta podstawowa tak skonstruowana, że bez problemu daje się zwiększyć możliwości komputera np. przez dołączenie dodatkowego układu Floating-Point-Coprocesor 68881 do niestandardowych operacji matematycznych. Są możliwe również inne rozszerzenia. Seryjnie system będzie wyposażony w szybki Block-Transfer-Chip (BLT) nazwany blitterem, którego zadaniem jest wspieranie TOS-u. MEGA ST jest w pełni kompatybilny z dotychczasowymi komputerami serii ST. Z tyłu posiada wejścia równoległe do drukarki, szeregowe RS232, do klawiatury, do dodatkowej stacji dysków, DMA do dysku twardego i drukarki laserowej, MIDI, VIDEO dla monitora kolorowego (mała i średnia rozdzielczość) i monochromatycznego (wysoka rozdzielczość). Także port ROM-u jest tak pomyślany, aby łatwo można było w przyszłości dołączyć do niego nowe rozszerzenia. Częścią systemu może być także dysk twardy 20 MB SH205. Zasilanie, kontroler i połączenie z komputerem są ukryte w obudowie. Przekazywanie danych odbywa się przez szybki DMA-Port (Direct Memory Acces). Przewidywane ceny w markach RFN:

ATARI MEGA ST 1 2498 DM
ATARI MEGA ST 2 2998 DM
ATARI MEGA ST 4 3998 DM

System wyposażony w monochromatyczny monitor i mysz.

Prawdopodobnie nie będzie już produkcji 260 ST, zamiast tego będzie w tej klasie 520ST/M z pojedynczą stacją dysków 3,5 cala i TOS-em w ROM-ie za 998 DM.

Ostatnią nowością pokazaną na CeBit 87' w Hanowerze przez firmę ATARI była drukarka laserowa SLM. Osiąga ona szybkość druku osiem stron na minutę i rozdzielczość 300 punktów na cal. Rozkazy przyjmuje bezpośrednio z komputera. Współpracuje tylko z komputerami serii MEGA ST od ST 2 w górę. Wyposażona jest w port DMA. Jest możliwość podłączenia do niej dodatkowych urządzeń. W sprzedaży razem z ATARI MEGA ST 2 za około 6000 DM.

W materiale wykorzystano informacje zawarte w miesięczniku ST COMPUTER.

Sergiusz Piotrowski



ATARI PC



ATARI MEGA ST 2

KLAN ATARI

CO JEST W ŚRODKU (3)

Układ GTIA (Graphics Television Interface Adaptor) jest także układem specjalnie zaprojektowanym dla komputerów Atari. Otrzymuje on dane obrazu od ANTIC-a, dodaje do nich informacje o kolorze i przesyła je następnie do monitora. Oprócz tego GTIA tworzy na ekranie ruchome obiekty czyli tzw. grafikę graczy i pocisków (Player/Missile Graphics — P/MG).

Podstawowym zadaniem GTIA jest uzupełnianie obrazu kolorem. Dla wypełnienia tej funkcji posiada on 9 rejestrów koloru. Są one jednakowo zbudowane: bity 1-3 określają jasność, a bity 4-7 barwę (bit 0 jest nie wykorzystany). Aby uzyskać odpowiedni kolor należy więc do takiego rejestru wpisać sumę wartości barwy pomnożonej przez 16 i jasności. Cztery rejestry przeznaczone są dla kolorów „pola gry" (COLor of Play Field), a cztery dla graczy i pocisków (COLor of Player/Missile). Oprócz tego istnieje jeszcze rejestr koloru tła (COLor of BAcKground). Kolor z tego rejestru jest wyświetlany zawsze tam, gdzie nie występują żadne dane obrazu, a więc w zasadzie na krawędziach obrazu. W niektórych trybach graficznych określa on także kolor ekranu (np. ANTIC 6-8 i in.).

GTIA umożliwia także uzyskanie trzech dodatkowych trybów graficznych. Są one sterowane bitami 6 i 7 rejestru GTIACNTL (zob. niżej). ANTIC pracuje wtedy w trybie 15, lecz dane obrazu przesłane do GTIA są przez niego inaczej interpretowane. Dane te są dzielone na czterobitowe kawałki, które służą do wyboru koloru w zależności od wartości pary bitów 7 i 6 w GTIACNTL. Gdy jest ona równa 01 (GRAPHICS 9), to kolor całego ekranu jest pobierany z rejestru COLBAK, a czterobitowe kawałki danych określają jasność punktów wyświetlanych na ekranie. Przy wartości 10 (GRAPHICS 10) kawałki te określają kolejne rejestry, z których pobierane są kolory punktów. Ponieważ rejestrów tych jest 9, to można uzyskać jednocześnie 9 kolorów. Para bitów 11 (GRAPHICS 11) działa podobnie jak 01, lecz z rejestru COLBAK pobierana jest jasność punktów, a wartość czterobitowego kawałka danych określa kolor. Umożliwia to uzyskanie 16 kolorów o jednakowej jasności (a więc nie można ich odróżnić w czarno-białym telewizorze).



Drugim bardzo ważnym zadaniem GTIA jest tworzenie grafiki graczy i pocisków. Ponieważ było to obszernie opisane w artykule „Duszki" w „Bajtku" nr 3/87, to ograniczę się tylko do kilku zagadnień, które zostały tam pominięte. GTIA tworzy jednorazowo jedną linię ekranu. W związku z tym część kształtu duszka, która ma się znaleźć w aktualnie wyświetlanej linii, jest chwilowo przechowywana w rejestrze grafiki (GRAFP, GRAFM). Zawartość tych rejestrów jest zmieniana co jedną linię ekranu przy rozdzielczości jednowierszowej i co dwie linie przy dwuwierszowej. Wyglądem „duszków" sterują również rejestry szerokości (SIZEP, SIZEM). W rejestrach SIZEP znaczenie mają tylko bity 0 i 1, a pozostałe są nie wykorzystane. W rejestrach GRAFM i SIZEM kolejne pary bitów odpowiadają kolejnym pociskom. W wielu programach istotne jest ustalenie, czy nastąpiła kolizja pomiędzy duszkami lub między duszkiem a polem gry. Do tego celu GTIA posiada rejestry kolizji, w których ustawienie jednego z bitów 0-3, oznacza, że nastąpiła kolizja z polem gry lub graczem (bity 4-7 są nie wykorzystane). Rejestry kolizji są rejestrami tylko do zapisu, a skasowanie ich zawartości następuje po wpisaniu dowolnej wartości do rejestru HITCLR.

Dodatkowym zadaniem GTIA jest kontrola przycisków joysticków i tzw. klawiszy kontroli czyli klawiszy START, SELECT i OPTION. Stan przycisków joysticków można sprawdzić w rejestrach TRIG0 i TRIG1. Wartość 1 oznacza przycisk zwolniony, a wartość 0 — naciśnięty. Klawisze konsoli obsługuje rejestr CONSOL, w którym bity są przyporządkowane klawiszom następująco: bit 0 — START, bit 1 — SELECT i bit 2 — OPTION. Bit skasowany oznacza, że odpowiedni klawisz jest wciśnięty. Bity 4-7 w tym rejestrze są nie wykorzystane, a bit 3 po skasowaniu wywołuje dźwięk („klick") klawiatury.

Teraz kolej na trochę adresów (O oznacza rejestr tylko do odczytu, a Z — tylko do zapisu).

53248–53251	HPOSP0-3	Poziome położenie graczy (Z)
53252–53255	HPOSM0-3	Poziome położenie pocisków (Z)
53256–53259	SIZEP0-3	Wielkość pozioma graczy (Z)
53260	SIZEM	Wielkość pozioma pocisków (Z)
		Wartości par bitów oznaczają odpowiednio: 00 — normalna wielkość (8 cykli koloru); 01 — podwójna wielkość (16 cykli); 10 — normalna wielkość; 11 — poczwórna wielkość (32 cykle)
53261–53264	GRAFP0-3	Rejestry grafiki graczy (Z)
53265	GRAFM	Rejestry grafiki pocisków (Z)
53248–53251	KOLM0-3PF	Kolizje pocisków z polem gry (O)
53252–53255	KOLP0-3PF	Kolizje graczy z polem gry (O)
53256–53259	KOLM0-3PL	Kolizje pocisków z graczami (O)
53260–53263	KOLP0-3PL	Kolizje graczy z graczami (O)
53278	HITCLR	Kasowanie rejestrów kolizji (Z)
53264–53265	TRIG0-1	Stan przycisków joysticków (O)
644–645	TRIG0-1§	Rejestry — cienie TRIG0-1
53266–53269	COLPM0-3	Kolory graczy i pocisków (Z)
704–707	COLPM0-3§	Rejestry — cienie COLPM0-3
53270–53273	COLPF0-3	Kolory pola gry (Z)
708–711	COLPF0-3§	Rejestry — cienie COLPF0-3
53274	COLBAK	Kolor tła (Z)
712	COLBAK§	Rejestr — cień COLBAK
53279	CONSOL	Stan klawiszy konsoli
53275	GTIACNTL	Rejestr kontroli GTIA. Bity 0-5 sterują P/MG, a bity 6 i 7 umożliwiają uzyskanie dodatkowych trybów graficznych
623	GTIACNTL§	
53277	PMCNTL	Rejestr kontroli „duszków"

Oprócz ANTIC-a i GTIA także system operacyjny ma swój udział w tworzeniu obrazu, ale o tym za miesiąc.

Wojciech Zientara

NIE BÓJ SIĘ PRZERWAŃ

—PŁYNĄCY NAPIS—

Czytelnicy „IKS-a", będący posiadaczami mikrokomputera Atari, zauważyli zapewne w numerze 2/87 tego czasopisma program „Komputerowy teleexpress".

Realizuje on uatrakcyjnienie posiadanych wcześniej programów przez dołączenie do nich strony tytułowej, czy też winiety. Ci Czytelnicy, którzy „wpalcowali" go na swoje Atari, stwierdzili z pewnością ograniczenia w wykorzystaniu tego programu, wynikające stąd, że został on napisany całkowicie w BASIC-u. Wyświetlany na ekranie tekst przeskakuje po literze, a nie jest płynnie przewijany. Ponadto w czasie, gdy program jest wykonywany, komputer nie może realizować innych działań, co jest istotnym mankamentem.

Program, którego wydruk znajduje się obok, jest pozbawiony tych dwóch niedoskonałości. Pracuje on w kodzie maszynowym 6502 wykorzystując dostępne na Atari przerwania. Prawdziwie płynny przesuw tekstu jest możliwy do osiągnięcia dzięki sprzętowemu wspomaganiu przewijania obrazu.

Ci Czytelnicy, którym zależy jedynie na posiadaniu programu, mogą w tym miejscu przerwać lekturę. Dla dociekliwych jeszcze kilka słów tytułem wyjaśnienia.

System operacyjny Atari wytwarza cały szereg różnorodnych przerwań. Najważniejszym z nich jest przerwanie synchronizacji pionowej (Vertical Blank Interrupt — w skrócie VBLK). Jest ono generowane po wykonaniu każdego obrazu (co 1/50 sekundy) i obsługuje m.in. procedury, które same nie mogą być źródłami przerwań, w tej liczbie również procedury obsługi zegarów ustawiane przez użytkownika. Były one opisane w „Bajtku" 4/87.

Licznikiem zegara TIMER2 są komórki pamięci o adresach 538 i 539. Gdy obie komórki zostaną wyzerowane, wykonywana jest procedura, której adres ustawia się w komórkach 552 oraz 553 (TIMER2VKT). Procedura ta musi się oczywiście kończyć rozkazem RTS, aby system nie ugrzązł w niej bezpowrotnie. W naszym programie TIMER2 wykonuje „czarną" robotę związaną z przesuwaniem pojawiającej się nad ekranem linii tekstu.

Możliwość płynnego przewijania linii obrazu o odcinki mniejsze niż wynikałoby to z mapy pamięci ekranu (w trybie graficznym 0 jeden bajt pamięci obsługuje obszar odpowiadający jednemu znakowi) jest również specyficzną cechą komputerów Atari. Linie obrazu, których rozkazy w programie ANTIC-a mają ustawiony 4 bit, są przesuwane w prawo o ilość punktów odpowiadającą zawartości rejestru HSCROLL — 54276. Nasz program zmieniając po każdym przerwaniu VBLK zawartość HSCROLL pozwala na uzyskanie opisanych efektów. Jednak sama modyfikacja zawartości tego rejestru pozwala na przesunięcie obrazu w zakresie kilku znaków. Aby uzyskać przesuwanie się większych partii tekstu trzeba połączyć płynny i zgrubny przesuw. Ten drugi realizowany jest przez modyfikację licznika pamięci obrazu ruchomej linii. Tekst wyświetlany w tej linii znajduje się, podobnie jak sam program, na szóstej stronie pamięci. Należy o tym pamiętać przy ewentualnych zmianach w programie.

Linia z przesuwającym się tekstem pojawia się po każdej instrukcji GRAPHICS. Również wciskanie klawisza RESET nie może usunąć jej z ekranu. Polecenie POKE 1560,96 zatrzymuje przesuw i dopiero wtedy GRAPHICS usuwa linię z ekranu. Pojawia się ona ponownie po wykonaniu POKE 1560,173.

Andrzej Biazik

```
0 REM ** 'Plynacy  napis' **
1 REM ** (c) 1987  A. Biazik **
2 REM zatrzymanie - POKE 1560,96
3 REM restart     - POKE 1560,173
10 S=0:FOR I=1536 TO 1643:READ A:POKE I
,A:S=S+A:NEXT I:IF S<>11583 THEN ? " BL
AD DANYCH":END
20 CLR :DIM A$(40):INPUT A$:? CHR$(125)
;A$
30 FOR I=1 TO 40:P=PEEK(39999+I):POKE 1
644+I,P:POKE 1684+I,P:NEXT I:? CHR$(125
)
40 POKE 12,1:POKE 13,6:POKE 9,1:X=USR(1
536):NEW
100 DATA 104,169,19,141,40,2,169,6,141,
41,2,169,4,133,208,169,40,133,207,169,1
,141,26,2
110 DATA 173,48,2,133,203,173,49,2,133,
204,160,2,177,203,201,112,208,25,56,165
,203,105
120 DATA 2,141,106,6,165,204,105,0,141,
107,6,169,100,141,48,2,169,6,141,49,2,1
98,208
130 DATA 165,208,141,4,212,208,23,169,4
,133,208,141,4,212,238,103,6,198,207,20
8,9,169
140 DATA 40,133,207,169,108,141,103,6,9
6,112,112,82,108,6,1,0,0
```

FORMATY RYSUNKÓW

Komputery Atari znane są ze swoich możliwości graficznych. Programy graficzne są najliczniej reprezentowane w grupie programów użytkowych na Atari. Dla tych, którzy lubią „grzebać w bitach" bardzo pomocna jest znajomość formatu, w jakim zapisywany jest obraz.

Najpopularniejszymi programami graficznymi na Atari XL/XE są Micropainter, Fun with Art i Koala Microillustrator. W tym artykule zostaną przedstawione formaty zapisu obrazu stosowane przez te programy.

MICROPAINTER

Jest to jeden z najstarszych i najprostszych programów. Używa trybu GRAPHICS 15. W trybie tym obraz złożony jest ze 192 linii po 40 bajtów, więc pamięć obrazu zajmuje 7680 bajtów. Wykonany rysunek jest zapisywany kolejno bajt po bajcie. Aby umożliwić odtworzenie kolorów, na końcu pliku zapisywane są zawartości rejestrów koloru w kolejności 712 (tło), 708, 709 i 710. Cały plik zawiera więc 7684 bajty

FUN WITH ART

Program ten używający także trybu GRAPHICS 15 umożliwia zmianę kolorów na rysunku poprzez zastosowanie przerwań programu ANTIC-a (DLI). Odtworzenie wykonanego rysunku wymaga więc zapisania w pliku linii, w których nastąpiły zmiany kolorów oraz tych kolorów. Format plików zapisywanych przez ten program jest następujący:

2 bajty — Nagłówek wskazujący, że jest to plik rysunku Fun with Art. Oba bajty są równe 254 ($FE).

4 bajty — Dane kolorów w takiej samej kolejności jak w MicroPainter (712, 708, 709, 710).

256 bajtów — Program ANTIC-a i tablica przerwań. Ten blok jest wczytywany na szóstą stronę pamięci (adresy 1536-1791).

4080 bajtów — Dane pamięci obrazu. Przy przekraczaniu granicy 4 KB musi nastąpić ponowne załadowanie licznika pamięci obrazu w układzie ANTIC. Funt with Art wpisuje więc dane w blokach.

16 bajtów — Zera wypełniające obszar pierwszego bloku pamięci obrazu do pełnych 4 KB (4096 B).

3600 bajtów — Drugi blok danych obrazu — uzupełniający obraz do 7680 bajtów.

2 lub więcej bajtów — Tu zapisane są dane dla przerwań programu ANTIC-a. Długość tego bloku zależy od ilości użytych przerwań i może wynosić maksymalnie 2304 bajty. Jeżeli nie zostało użyte żadne przerwanie, to oba bajty są równe zero.

KOALA MICROILLUSTRATOR

Najbardziej znany program graficzny, posiadający także wersję na cartridge'u. Korzysta również z trybu 15. W celu zaoszczędzenia nośnika program ten stosuje specjalny, skondensowany format zapisu plików. Oprócz tego została w nim uwzględniona możliwość przyszłej modyfikacji i rozbudowy programu. W związku z tym plik zapisywany przez niego posiada duży nagłówek precyzujący parametry obrazu. Format tego nagłówka jest następujący:

4 bajty — Identyfikator wskazujący, że jest to plik Koala. Bajty te są równe 255, 128, 201 i 199.

2 bajty — Długość nagłówka: liczba bajtów nagłówka minus jeden zapisana jako młodszy i starszy bajt. Zwykle bajty te są równe 26 i 0.

1 bajt — Numer wersji programu, teraz równy 1.

1 bajt — Rodzaj zastosowanej kompresji rysunku: 0 — nieskondensowany, 1 — kompresja pionowa, 2 — kompresja pozioma.

1 bajt — Numer trybu graficznego ANTIC-a (UWAGA! różny od numeru trybu GRAPHICS). Normalnie równy 14. Zapisywanie tej wartości świadczy o tym, że przewidywane były wersje programu używające innych trybów.

4 bajty — Aktualna konfiguracja pamięci obrazu — ilość bajtów w linii (0 i 40) oraz ilość linii (0 i 192). Prawdopodobnie przewidywane do zastosowania w dalszych wersjach.

5 bajtów — Dane kolorów w kolejności rejestrów od 708 do 712.

2 bajty — Całkowita długość pliku obrazu.

2 bajty — Oba równe zero. Przewidziane do wykorzystania w przyszłości.

1 bajt — Obecnie zawiera wartość 155 (RETURN). Jest to miejsce przewidziane do zapisania tytułu rysunku (do 14 znaków); aktualnie nie używane.

1 bajt — 155 (RETURN). Miejsce na nazwisko autora (14 znaków); aktualnie nie używane.

2 bajty — Oba równe 155. Przewidziane na dwie linie tekstu, obecnie nie używane.

1 bajt — Bajt odstępu oznaczający koniec nagłówka. Równy 162.

Po nagłówku następują dane obrazu zapisane w postaci skondensowanej. Długość pliku zależy od szczegółowości rysunku, a dokładniej od liczby powtarzających się danych. Przed zapisaniem pliku program przeszukuje pamięć obrazu i zapisuje dane w blokach. Pierwszy bajt każdego bloku określa jego rodzaj. Stosowane są cztery rodzaje bloków o następujących formatach:

1. Pierwszy bajt 0nnnnnnn, gdzie nnnnnnn jest różne od zera i oznacza ilość powtórzeń danej. Drugi bajt zawiera daną, która ma być powtarzana. Umożliwia zapisanie w dwóch bajtach danej powtarzającej się do 127 razy.
2. Pierwszy bajt 00000000. Dwa następne bajty zawierają liczbę powtórzeń danej (od 0 do 65535). Czwarty bajt jest daną, która ma być powtarzana.
3. Pierwszy bajt 1nnnnnnn, gdzie nnnnnnn jest różne od zera i oznacza ilość bajtów danych w bloku (0-127). Drugi i następne bajty zawierają niepowtarzalne dane obrazu.
4. Pierwszy bajt 10000000. Drugi i trzeci bajt określają ilość bajtów danych (0-65535). Dalsze bajty zawierają niepowtarzalne dane obrazu.

ZASTOSOWANIE

Dobrze, tylko po co to wszystko? Programów zamieniających rysunki z jednego formatu na inny jest kilka (sam znam cztery), więc nikt nie będzie się w to bawił. To prawda, lecz zastosowań dla podanych tu informacji można znaleźć wiele. Oto najprostsze: napisałeś grę i chciałbyś ozdobić ją ładnym obrazkiem tytułowym. Narysować nie problem, ale jak potem wczytać skoro plik MicroPaintera ma prawie 8 KB (ile to będzie trwało z magnetofonu!). Rozwiązanie samo się narzuca: plik Koali jest 4-5 razy krótszy, trzeba tylko napisać procedurę, która go odczyta. Oszczędność czasu będzie mniej więcej trzykrotna. A ile satysfakcji!

Wojciech Zientara

ZAMIANA NAPISÓW W PROGRAMACH

Prawie wszystkie programy dostępne dla mikrokomputerów pochodzą z zagranicy. Z tego powodu wszystkie napisy w takich programach są obcojęzyczne. Oczywiście każdy, kto choć raz miał do czynienia z komputerem, wie co znaczy SCORE. Ale dlaczego nie miałoby być napisane WYNIK.

Poniższy program umożliwia skopiowanie programu z kasety lub dyskietki, wymianę napisów i ponowne zapisanie na kasecie lub dyskietce.

Działanie programu jest bardzo proste. Na początku należy podać nazwę zbioru źródłowego i docelowego (dla magnetofonu C, dla stacji dysków D:NAZWA). Program nasz najpierw odczytuje kopiowany program traktując go jako ciąg znaków. Następnie żąda podania tekstu, który ma być usunięty, i tekstu, który ma być wstawiony na jego miejsce. Muszą one być tej samej długości — gdy nowy tekst jest dłuższy, trzeba użyć skrótów; gdy krótszy, należy uzupełnić go spacjami (na początku). Teraz komputer wyszukuje we wczytanym programie stary tekst i zamienia go na nowy. Dokonana zamiana jest sygnalizowana dźwiękiem, równocześnie wyświetlany jest numer kolejny bajtu, od którego zaczyna się wymieniany tekst. Po przejrzeniu całego programu ponownie następuje żądanie podania tekstu do wymiany i procedura jest powtarzana. Aby zakończyć program należy w odpowiedzi na żądanie tekstu do wymiany nacisnąć klawisz RETURN. Jeszcze tylko zapisanie poprawionego programu i możemy korzystać z polskiej wersji.

Na zakończenie trzy praktyczne uwagi:

1. Teksty trzeba podawać dokładnie. Jeżeli w programie jest HI-SCORE, a podamy HISCORE, to napis nie zostanie zmieniony.
2. Przed wymianą napisów trzeba zastanowić się nad kolejnością. Np. podajemy stary tekst SCORE i nowy WYNIK. Gdy teraz podamy Tekst HI-SCORE, to nie będzie on zmieniony, bo w tym miejscu jest już HI-WYNIK.
3. Należy unikać krótkich napisów, gdyż można spowodować zmiany programu. Np. tekst HI odpowiada sekwencji rozkazów PHA, EOR n, a tekst LV — sekwencji JMP 86+256 ∗ n.

Wojciech Zientara

```
10 DIM F1$(20),F2$(20),ML$(28),S1$(30),
S2$(30):POKE 710,0:TRAP 120:OPEN #5,4,0
,"K:"
20 RAM=FRE(0)-4000:DIM D$(RAM+1)
30 ? "[CLEAR][DOL]*[Y]BFCH 2.0[Y]*  Boo
t-File-Changer 2.0":? "[DOL] Copyright
(c) 1986 by W. Zientara[2 DOL]"
40 ? "   INPUT-FILE ";:INPUT F1$:IF F1$
(1,1)="C" THEN A1=128
50 ? "[DOL]  OUTPUT-FILE ";:INPUT F2$:I
F F2$(1,1)="C" THEN A2=128
60 CLOSE #1:OPEN #1,4,A1,F1$:? "Czytam
";F1$
70 ML$="hhh*hh[INS]B[C]h[INS]E[C]h[INS]
D[C]h[INS]I[C]h[INS]H[C]LVd"
80 X=USR(ADR(ML$),16,7,ADR(D$),RAM):L=P
EEK(856)+256*PEEK(857)
90 ST=PEEK(851):IF ST=1 OR ST=136 THEN
130
100 ? "[2] BLAD  ";ST
110 FOR I=1 TO 1000:NEXT I:RUN
120 ST=PEEK(195):? "[DOL]W LINII ";PEEK
(186)+256*PEEK(187);" ";:GOTO 100
130 D$(L+1)=""
140 ? "Zbior zaladowany w calosci.":CLO
SE #1
150 ? "Blok danych ma dlugosc ";L;" baj
tow":FOR I=1 TO 1000:NEXT I
160 POSITION 2,5:? "[12 DEL][DOL]":POKE
752,0
170 ? "PODAJ STARY TEKST (max.30 znakow
)":INPUT S1$:IF S1$="" THEN 230
180 ? "PODAJ NOWY TEKST (tej samej dlug
osci)":INPUT S2$:IF LEN(S2$)<>LEN(S1$)
THEN ? "[2][2 GORA][3 DEL]":GOTO 180
190 K=LEN(S1$)-1:POKE 752,1:FOR I=1 TO
L-K
200 IF D$(I,I+K)=S1$ THEN D$(I,I+K)=S2$
:? "[2] ";I;:I=I+K
210 NEXT I:GOTO 160
230 ML$="hhh*hh[INS]B[C]h[INS]E[C]h[INS
]D[C]h[INS]I[C]h[INS]H[C]LVd":? "[2 GOR
A][3 DEL]";
250 CLOSE #1:OPEN #1,8,A2,F2$:? "Zapisu
je ";F2$
260 X=USR(ADR(ML$),16,11,ADR(D1$),L):IF
PEEK(851)<>1 THEN 100
270 CLOSE #1:END
```

KANAŁY I STRUMIENIE — cz. I

Kanały i strumienie mogą być wykorzystywane zarówno przez tych, którzy swe programy piszą tylko w BASIC-u, jak i tych, którym nieobcy jest asembler. Pisząc programy całkowicie w BASIC-u nie można wprawdzie wykorzystać wszystkich możliwości, które kanały oferują użytkownikowi, lecz w niektórych przypadkach można uzyskać bardzo ciekawe efekty.

Idea w wyniku której powstały kanały jest bardzo prosta: maksymalnie ujednolicić sposób wysyłania informacji do urządzeń peryferyjnych, takich jak drukarka, ekran, microdrive itp. — wszystkie te informacje wysyłać za pomocą jednej instrukcji. W BASIC-u jest nią właśnie PRINT.

Kanał jest symulacją linii przesyłowej łączącej komputer z którymś z tych urządzeń. Chcąc przykładowo przekazać informacje drukarce, należy przesyłać je używając PRINT-a, lecz za pomocą odpowiedniej „linii" — kanałem służącym do porozumiewania się z drukarką.

Normalnie w SPECTRUM istnieją 4 kanały:

Kanał „P" PRINTER — służący do wysyłania informacji na drukarkę
Kanał „S" SCREEN — wykorzystywany do druku w górnej części ekranu
Kanał „K" KEYBOARD — służący do wysyłania oraz przyjmowania (np. w instrukcji INPUT, podczas wprowadzania komend z klawiatury itp.) informacji, przy wykorzystaniu dolnej części ekranu
Kanał „R" WORKSPACE — zupełnie nieprzydatny dla BASIC-a, a służący do magazynowania napływających informacji, które potem mogą być odczytane wszystkie razem.

Z poziomu BASIC-a można korzystać tylko z trzech pierwszych kanałów, czyli K, S i P.

Aby wyprowadzić z komputera informacje w którymś z tych kanałów, należy posłużyć się nie jego nazwą, lecz numerem. Numer ten, to tzw. strumień. Przyporządkowanie kanałów strumieniem jest następujące:

Strumienie −3,0,1	—	kanał K
−2,2	—	S
3	—	P
−1	—	R

Strumienie 4 ÷ 15 są początkowo zamknięte, tzn. nie jest im przypisany żaden kanał.
Wszystkie strumienie ujemne są dla BASIC-a niedostępne, lecz aby skorzystać z kanałów K, S i P wystarczą strumienie 0,1,2 i 3.

Instrukcja PRINT drukuje w kanale S (strumień 2). LPRINT, które daje wydruk na drukarce, różni się od PRINT tylko tym, że korzysta z kanału P (poprzez strumień 3). Jeżeli chcemy wewnątrz instrukcji PRINT lub LPRINT zmienić kanał wydruku, to musimy użyć znaku # (hash), po którym występuje numer wybranego przez nas strumienia. Jeśli napiszemy:

PRINT # 3;„tekst"

to będzie to równoznaczne z LPRINT „tekst". Napis pojawi się na drukarce. Kanał wydruku można także zmieniać kilkakrotnie w jednej instrukcji:

LPRINT „to jest na drukarce" # 2;„to na górze ekranu"; # O „a to na dole ekranu": PAUSE 0

Instrukcja PAUSE jest tu po to, by napis, który się pojawi w dolnej części ekranu nie został od razu skasowany przez komunikat „0 O.K.".

Teoretycznie istnieje także możliwość wyboru kanału w instrukcji INPUT, lecz wybór ten ogranicza się tylko do dwóch strumieni: 0 i 1, czyli tylko do kanału K. Działanie INPUT-a w innych kanałach nie jest możliwe, ponieważ tylko kanał K może służyć jako kanał wejściowy. W tym przypadku są one odbierane z klawiatury oraz wysyłane na ekran, do dolnej części ekranu.

Oprócz PRINT #, INPUT # i LIST #, istnieją jeszcze dwie instrukcje, pozwalające otwierać lub zamykać strumienie przeznaczone dla użytkownika (strumienie 4 ÷ 15). Są to OPEN # które strumień otwiera, oraz CLOSE #, które go zamyka.

OPEN # nr, nazwa kanału — otwiera strumień o podanym numerze przyporządkowując mu kanał o podanej nazwie. Nazwa ta jest jednoznakowym łańcuchem „K", „S" lub „P" (albo „k", „s", „p"). Może to być oczywiście zmienna łańcuchowa, lecz gdy nie jest ona żadnym z tych łańcuchów lub jej długość jest większa niż jeden znak, to jest generowany komunikat „Invalid file name".

Jeśli więc np. chcemy otworzyć strumień 4 dla kanału K, to piszemy: OPEN # 4, „K" lub OPEN # 4, „k" i od tej pory PRINT # 4;... drukuje w dolnej części ekranu. Dzieje się tak do momentu, w którym zamkniemy strumień 4 przez CLOSE # 4. Teraz próba użycia tego strumienia wywoła jedynie protest naszego komputera w postaci komunikatu „Invalid stream". Przy zamykaniu kanałów należy bardzo uważać, ponieważ na skutek błędu w ROM-ie próba zamknięcia kanału, który został już zamknięty wcześniej lub w ogóle nie był otwierany, kończy się zazwyczaj zawieszeniem się komputera.

W programach nie wykorzystujących w ogóle procedur w języku maszynowym, kanały mogą być użyte w programach drukujących jakieś dane (np. w przedstawionym niżej przykładzie, tablice a i b$) w zależności od potrzeb, na ekranie lub drukarce:

```
  10 DIM a(20): DIM b$(20,16)
    ...
 100 OPEN #4,"s": GO SUB 1000
 110 OPEN #4,"p": GO SUB 1000
    ...
1000 REM podprogram drukujący tabl
ice a() i b$()
1010 PRINT #4;"    Tablica a","Tabl
ica b$"
1020 FOR n=1 TO 20
1030 PRINT #4;n;TAB 3;a(n),b$(n)
1040 NEXT n
1050 RETURN
```

To jest praktycznie wszystko, do czego można wykorzystać kanały posługując się wyłącznie BASIC-iem. Szersze ich wykorzystanie jest możliwe dopiero przy pomocy asemblera i procedur umieszczonych w ROM-ie. Nie znaczy to jednak, że w BASIC-u niczego więcej już się nie da zrobić — można przecież napisaną w asemblerze procedurę obsługi kanału włączyć do programu w BASIC-u umieszczając ją w liniach DATA lub wczytując ją z taśmy przez LOAD "" CODE. Po wpisaniu kilku instrukcji POKE „podłączających" naszą procedurę do istniejących kanałów można już z niej korzystać przez PRINT, LIST czy INPUT. Tym, jak to dokładnie robić zajmiemy się w drugiej części tego artykułu, za miesiąc.

W tym języku drukowanie jest trochę bardziej skomplikowane, nie ma bowiem instrukcji odpowiadającej dokładnie PRINT-owi. Podczas działania programu, po rozpoznaniu przez interpreter BASIC-a tej instrukcji, wykonywany jest szereg procedur, a nie jakaś jedna załatwiająca od razu wszystko. Pierwszą z nich jest CHAN-OPEN znajdująca się pod adresem 5633, a służąca do otwarcia kanału dla wydruku. Otwarcie nie oznacza tu tego samego co w BASIC-u, gdyż tam było to przypisanie strumieniowi kanału — tutaj jest to tylko znalezienie i zapamiętanie kanału o podanym numerze. Przed wywołaniem tej procedury w akumulatorze umieszczamy numer otwieranego kanału. Liczba ta jest zapisana w kodzie U2. tzn. gdy jest ujemna lecz większa od -129, to aby uzyskać jej zapis w tym kodzie dodajemy do niej 256, a gdy dodatnia lecz mniejsza od 128, to piszemy ją tak jak w zapisie dziesiętnym.

Jeśli otworzyliśmy już kanał, to możemy zabrać się do drukowania. Podstawową procedurą, wyprowadzającą pojedynczy bajt informacji w używanym właśnie kanale jest podprogram umieszczony pod adresem 16, który można wywoływać przez CALL 16, lub prościej przez RST 16. Procedura ta wysyła w otwartym wcześniej kanale znak, którego kod znajduje się w akumulatorze. Aby więc np. wydrukować w górnej części ekranu literę „A" możemy wpisać taki program: (jeśli nie masz wczytanego asemblera, to użyj programu przedstawionego na końcu.

```
3E FE       LD A,-2    ;numer strumienia
CD 01 16    CALL 5633  ;otwarcie kanału S
3E 41       LD A,65    ;kod litery A
D7          RST 16     ;wydruk "A"
C9          RET        ;powrót
```

Najpierw otwieramy kanał przypisany strumieniowi -2 czyli S, a potem wysyłamy w nim kod litery „A". Jeśli chcemy teraz wydrukować w kanale S następny znak, to nie musimy go ponownie otwierać. Każdy kanał pozostaje otwarty aż do momentu gdy otworzy się inny kanał.

Jeśli chcemy wydrukować jakiś dłuższy tekst, to wysyłanie każdego znaku osobno nie jest zbyt wygodne. Na całe szczęście w ROM-ie znajduje się kilka podprogramów, które mogą nas w tym wyręczyć. Pierwszym z nich jest podprogram umieszczony pod adresem 8252. Drukuje on tekst, którego długość podajemy w parze rejestrów BC, a adres pierwszego znaku w parze HL DE. Tekst ten może składać się z dowolnych znaków o kodach od 0 do 255. Ponieważ wszystkie te podprogramy mogą działać w różnych kanałach, więc przed wywołaniem każdego z nich musimy pamiętać o otwarciu odpowiedniego kanału

```
3E 02       LD A,2      ;numer strumienia
CD 01 16    CALL 5633   ;otwarcie kanału S
11 A2 09    LD DE,2466  ;adres łańcucha
01 1E 00    LD BC,30    ;długość
CD 3C 20    CALL 8252   ;wydruk napisu
C9          RET         ;powrót
```

Drugą z procedur drukujących teksty jest procedura umieszczona pod adresem 3082. Drukuje ona tylko łańcuchy złożone ze znaków o kodach 0 ÷ 127 lecz za to możemy umieścić w pamięci jeden za drugim do 256 łańcuchów nie martwiąc się o adres ani długość każdego z nich.

W bajcie pamięci tuż przed pierwszym łańcuchem, umieszczamy liczbę większą niż 127, a za nią tekst pierwszego łańcucha. Aby zasygnalizować gdzie jest jego koniec, do kodu jego ostatniego znaku dodajemy 128. Zaraz po tym bajcie może znajdować się początek następnego łańcucha zapisanego w identyczny sposób. Chcąc teraz wydrukować któryś z tych tekstów, w parze rejestrów DE umieszczamy adres bajtu poprzedzającego pierwszy tekst, a w akumulatorze podajemy, który z nich ma być wydrukowany. Jest to liczba z zakresu 0 ÷ 255, gdzie 0 oznacza pierwszy łańcuch, 1 — drugi itd. W ROM-ie znajduje się kilka bloków takich „upchniętych" tekstów. Spróbujmy wyświetlić zawartość jednego z nich, zawierającego wszystkie komunikaty o błędach:

```
3E 02       LD A,2      ;numer strumienia
CD 01 16    CALL 5633   ;otwarcie kanału S
AF          XOR A       ;wyzerowanie rej. A
F5          PUSH AF     ;zapamiętanie stanu A
11 91 13    LD DE,5009  ;adres bajtu przed
                        ; pierwszym tekstem
CD 0A 0C    CALL 3082   ;druk tekstu o num. A
3E 0D       LD A,13     ;przejście do nowej
D7          RST 16      ; linii (druk ENTER)
F1          POP AF      ;odtworzenie stanu A
3C          INC A       ;następny tekst
FE 1E       CP 30       ;powrót jeśli był
D0          RET NC      ; już 30-ty tekst
18 EF       JR -17      ;skok do PUSH AF
```

Przeanalizuj działanie tego programu i porównaj wydrukowane napisy z zawartością obszaru 5009—5460 (sprawdź to przez PEEK... i CHR$ PEEK...)

Jeżeli w A znajduje się liczba od 0 do 9, to możesz ją wydrukować przez CALL 5615 — jest to równoznaczne z ADD,A,48 (kod znaku „0") przez RST 16. Jeśli zaś chcesz wydrukować liczbę większą ale mniejszą niż 10 000, to możesz ją umieścić w parze BC oraz wykonać podprogram znajdujący się pod adresem 6683.

Jeśli i taki zakres ci nie wystarcza, to musisz posłużyć się procedurami tzw. „kalkulatora". Jest to ta część interpretera BASIC-a, która zajmuje się wszelkimi obliczeniami, a do przechowywania liczb wykorzystuje specjalny obszar pamięci — „CALCULATOR STACK" — czyli „stos kalkulatora". Za pomocą procedur kalkulatora można trzymane na stosie liczby dodawać, odejmować, mnożyć, potęgować, traktować jako argumenty różnych funkcji itp. Można także wydrukować wartość liczby znajdującej się na szczycie tego stosu. Nam wystarczą tylko dwa podprogramy: pierwszy — który zapisze liczbę z rejestrów mikroprocesora na szczyt stosu, drugi — który ją wydrukuje.

Aby wpisać na liczbę na stos, umieszczamy ją w parze rejestrów BC oraz wywołujemy podprogram o adresie 11563. Teraz otwieramy odpowiedni kanał i drukujemy zapisaną na stosie liczbę przez CALL 11747:

```
01 39 30    LD BC,12345 ;zapisanie liczby
CD 2B 2D    CALL 11563  ; 12345 na stos kal.
3E 02       LD A,2      ;numer strumienia
CD 01 16    CALL 5633   ;otwarcie kanału S
CD E3 2D    CALL 11747  ;druk liczby ze szczy-
C9          RET         ; tu stosu i powrót
```

Te podprogramy umożliwiają wydrukowanie praktycznie dowolnego tekstu czy liczby w wygodny dla piszącego program w asemblerze sposób. Są one odpowiednikami instrukcji PRINT na poziomie asemblera. Te procedury są konieczne aby móc korzystać z kanałów w asemblerze oraz aby zrozumieć w jaki sposób za pomocą własnych programów zaprząc kanały do jakiejś pożytecznej pracy, np. ulepszać działanie PRINT-a w BASIC-u lub nawet zabezpieczać swe programy przed wylistowaniem, ale o tym w drugiej części, za miesiąc.

Tomasz Surmacz

Program umożliwiający uruchomienie przykładowych programów w asemblerze:

```
  1 REM uruchamianie przykładów w
       asemblerze
 10 CLEAR 59999: LET x=60000
 20 READ a,b,c,d,e,f
 30 DATA 10,11,12,13,14,15
 40 READ a$
 50 FOR n=1 TO LEN a$ STEP 2
 60 POKE x,VAL a$(n)*16+VAL a$(n+1)
 70 LET x=x+1: NEXT n
 80 RANDOMIZE USR 60000
 85 REM umieść kod szesnastkowy
    programu w linii 90 DATA "..."
```

np: 90 DATA „3EFECD01163E41D7C9"

EDYTOR ZNAKÓW GRAFICZNYCH

Prezentowany program umożliwia szybkie zaprojektowanie zestawu 96 znaków o kodach ASCII 32 do 127 i znaków graficznych użytkownika.

Po uruchomieniu programu przez RUN należy odczekać chwilę, aż w odpowiednim obszarze RAM umieszczone zostaną procedury maszynowe. Następnie pojawi się pytanie czy wczytać dane z taśmy magnetofonowej, a później plansza prezentująca poprawiany zestaw znaków i ich odpowiedniki w kodzie ASCII, menu oraz edytowany znak w wielkości naturalnej i powiększony 8 razy. W programie możliwa jest realizacja kilkunastu funkcji wybieranych naciśnięciem odpowiedniego klawisza.

WYŚWIETLANIE (klawisze 1—8) — poprawianie dowolnego wiersza aktualnie wyświetlanego znaku. Poprawkę podajemy w postaci tekstu, w którym spacje oznaczają punkt w kolorze tła, a wszystkie pozostałe znaki zamieniane są na punkty w kolorze papieru.

OBRÓT W LEWO (klawisz L) — obrócenie aktualnie wyświetlanego znaku w lewo.

OBRÓT W PRAWO (klawisz R) — obrócenie aktualnie wyświetlanego znaku w prawo.

ODBICIE LUSTRZANE (klawisz O) — odbicie lustrzane aktualnie wyświetlanego znaku.

ODBICIE Z OBROTEM (klawisz P) — przekształcenie złożone z odbicia i obrotu.

KODOWANIE (klawisz K) — zakodowanie aktualnie wyświetlanego znaku w edytowanym zestawie.

SUMA (klawisz S) — zsumowanie aktualnie wyświetlanego znaku z dowolnym innym z wyjątkiem znaków o kodach ASCII od 128 do 143.

ZAPIS (klawisz Z) — zapisanie utworzonego zbioru 96 znaków na taśmie magnetofonowej.

PRZESUNIĘCIE W LEWO (klawisz X) — przesunięcie w lewo aktualnie wyświetlanego znaku. Lewa kolumna punktów jest kasowana.

PRZESUNIĘCIE W PRAWO (klawisz M) — przesunięcie w prawo aktualnie wyświetlanego znaku. Prawa kolumna punktów jest kasowana.

PRZESUNIĘCIE W GÓRĘ (klawisze LXR) — przesunięcie w górę aktualnie wyświetlanego znaku zrealizowane przez wykonanie trzech poprzednio opisanych funkcji.

PRZESUNIĘCIE W DÓŁ (klawisze LMR) — przesunięcie w dół aktualnie wyświetlanego znaku.

Utworzony zestaw 96 znaków graficznych może zastąpić znaki standardowe, które zapisane są w pamięci ROM. Należy umieścić go w wybranym miejscu pamięci RAM wykonując rozkazy:

CLEAR adr-1: LOAD „nazwa" CODE adr, 768

Po wykonaniu tej czynności musimy zmienić wartość zmiennej systemowej CHARS, która informuje system o położeniu generatora znaków w pamięci. Realizujemy to przez:

POKE 23606, l : POKE 23607,h

Wartości l i h obliczamy ze wzorów l =x − 256 ∗ INT (X/256) oraz h = INT (X/256) gdzie X = adr − 256.

Program umieszcza dane o edytowanych znakach od adresu adr = 64472. Efekt naszej pracy widoczny jest na ekranie po wpisaniu POKE 23606,216 : POKE 23607,250 (linia 6540). Powrót do standardowego generatora następuje po wykonaniu POKE 23606,0 : POKE 23607,60. Podczas testowania edytora utworzono zestaw pogrubionych liter i cyfr, które znacznie poprawiają czytelność wyświetlanych tekstów.

Program składa się z czterech zasadniczych części:

1. linie 1—112 — wczytanie procedur maszynowych
2. linie 120—200 — inicjalizacja zmiennych, przepisanie generatora znaków z ROM, wczytanie danych z taśmy, wydruk planszy
3. linie 210—300 — główna pętla sterująca programu
4. linie 4999—7550 — procedury realizujące poszczególne funkcje.

Użytkownik może łatwo wzbogacić program o własne procedury dodając w pętli głównej wywołanie kolejnej opcji. Proponuję samodzielne dopisanie procedury edycji znaku za pomocą manipulatora. Nie wolno zmieniać linii 6545, gdyż wszystkie przekształcenia i wyświetlanie wymaga uprzedniego wydrukowania znaku przez PRINT AT 17,21 ; „Z$". Poprawianie odbywa się przez bezpośrednie wpisywanie odpowiednich wartości do pamięci obrazu przez POKE np.: linia 6430 lub pętla 7150—7170.

Janusz Jarmoch

```
1
   1 REM EDYTOR ZNAKÓW GRAFICZNYCH
   3 CLEAR 64463: LET A=65239
   9 REM PROCEDURY MASZYNOWE
  10 PRINT AT 10,10;"***ZACZEKAJ***"
  20 FOR K=100 TO 112
  30 LET S=0
  40 FOR L=1 TO 10
  50 LET A=A+1: READ B: POKE A,B: LET S=S+B
  60 NEXT L
  70 READ SK: IF S<>SK THEN  PRINT "BŁĄD W ";K
  80 NEXT K
 100 DATA 33,53,80,17,208,250,229,205,57,255,1387
 101 DATA 225,14,8,221,33,208,250,6,8,221,1194
 102 DATA 203,0,46,203,22,221,35,16,246,36,1028
 103 DATA 13,32,236,33,53,80,17,7,89,14,574
 104 DATA 8,6,8,175,62,63,203,22,48,2,597
 105 DATA 62,0,18,19,16,244,203,22,36,62,682
 106 DATA 24,131,95,13,32,231,201,33,0,61,821
 107 DATA 17,216,251,1,0,3,237,176,201,33,1135
 108 DATA 0,0,17,216,250,41,41,41,25,17,648
 109 DATA 53,80,235,201,205,39,255,6,8,126,1208
 110 DATA 18,36,19,16,250,201,205,39,255,6,1045
 111 DATA 8,26,79,126,177,119,36,19,16,247,853
 112 DATA 24,167,118,27,3,19,0,62,0,60,480
 120 DIM B$(8)
 140 DIM W$(96)
 150 FOR K=1 TO 96: LET W$(K)=CHR$ (K+31): NEXT K
 160 RANDOMIZE USR 65307
 170 PRINT #0;"CZY WCZYTAĆ DANE Z TAŚMY ? T-N "
 180 GO SUB 6010
 190 IF S$="T" THEN  GO SUB 7300
 195 CLS
 200 GO SUB 5000: GO SUB 5100: REM WYDRUK PLANSZY
 210 REM  PĘTLA GŁÓWNA
 220 GO SUB 6000
 230 IF S$>"0" AND S$<"9" THEN  GO SUB 6400
 240 IF S$="W" THEN  GO SUB 6500
 245 IF S$="K" THEN  GO SUB 6600
 250 IF S$="L" THEN  GO SUB 6700
 255 IF S$="X" THEN  GO SUB 7150
 260 IF S$="R" THEN  GO SUB 6800
 265 IF S$="P" THEN  GO SUB 6830
 270 IF S$="O" THEN  GO SUB 6790
 275 IF S$="S" THEN  GO SUB 7000
 290 IF S$="Z" THEN  GO SUB 7500
 295 IF S$="M" THEN  GO SUB 7100
 300 GO TO 220
4999 REM WYDRUK GENERATORA ZNAKÓW
5000 LET M=1
5010 POKE 23606,216: POKE 23607,250: GO SUB 5030
5015 POKE 23606,0: POKE 23607,60
5020 LET M=0
5030 PRINT AT 0+M,0;W$( TO 32);AT 2+M,0;W$(33 TO 64)
5035 PRINT AT 4+M,0;W$(65 TO )
5040 RETURN
5100 REM PLANSZA
5110 FOR K=1 TO 8
5120 PRINT AT 7+K,5;K;AT 17,6+K;K
5130 NEXT K
5135 PLOT 55,112: DRAW 65,0: DRAW 0,-65
5136 DRAW -65,0: DRAW 0,65
5150 PRINT AT 16,20;"▛▀▜";AT 17,20;"▌ ▐";AT 18,20;"▙▄▟
  "
5160 PRINT AT 8,16;"WYŚWIETLANIE...W"
5170 PRINT AT 9,16;"POPRAWIANIE..1-8"
5180 PRINT AT 10,16;"OBRÓT........L.R"
5190 PRINT AT 11,16;"ODBICIE......O.P"
5200 PRINT AT 12,16;"KODOWANIE......K"
5210 PRINT AT 13,16;"SUMA...........S"
5220 PRINT AT 14,16;"ZAPIS..........Z"
5230 PRINT AT 15,16;"PRZESUNIĘCIE.X.M"
5250 RETURN
5999 REM WCZYTAJ ZNAK Z KLAWIATURY
6000 PRINT #0;" WYBIERZ POZYCJĘ Z MENU "
6010 LET S$=INKEY$: IF S$<" " THEN  GO TO 6010
6020 IF S$>="a" THEN  LET S$=CHR$ (CODE S$-32)
6025 INPUT ""
6030 RETURN
6399 REM  POPRAW WIERSZ
6400 INPUT "       12345678                  ";B$
6403 LET S=0
6405 FOR K=1 TO 8
6410 LET M=0: IF B$(K)<>" " THEN  LET M=1
6415 LET S=S*2+M
6420 NEXT K
6430 POKE 20533+256*(VAL S$-1),S
6450 RANDOMIZE USR 65273
6460 RETURN
6500 REM  WYŚWIWETLANIE ZNAKU
6510 INPUT "JAKI ZNAK WYŚWIETLIĆ ? "; LINE Z$
6520 LET Z=CODE Z$
6530 IF Z<32 OR Z>164 THEN  GO TO 6510
6540 POKE 23606,216: POKE 23607,250
6545 PRINT AT 17,21;CHR$ Z: POKE 23606,0: POKE 23607,60
6550 RANDOMIZE USR 65273
6560 RETURN
6600 REM KODOWANIE
6610 INPUT "JAKI ZNAK ZAKODOWAĆ ? "; LINE Z$: LET Z=CODE Z$
6620 IF Z<32 OR (Z>127 AND Z<144) OR Z>164 THEN  RETURN
6630 POKE 65320,Z
6640 RANDOMIZE USR 65334
6650 GO SUB 5000
6660 RETURN
6699 REM  PRZEKSZTAŁCENIA
6700 POKE 65262,46: POKE 65264,22
6710 GO TO 6950
6790 GO SUB 6840
6800 POKE 65262,38: POKE 65264,30
6810 GO TO 6950
6840 POKE 65262,38: POKE 65264,22
6950 RANDOMIZE USR 65240
6960 RETURN
6999 REM SUMOWANIE 2 ZNAKOW
7000 INPUT`" JAKI ZNAK DOPISAĆ ? "; LINE Z$: LET Z=CODE Z$
7010 IF Z<32 OR Z>127 AND Z<144 OR Z>164 THEN  RETURN
7020 POKE 65320,Z: RANDOMIZE USR 65346
7030 RETURN
7099 REM PRESUNIĘCIE W PRAWO
7100 FOR K=20533 TO 22325 STEP 256
7110 LET L=INT (PEEK K/2): POKE K,L
7120 NEXT K: RANDOMIZE USR 65273
7130 RETURN
7149 REM PRZESUNIĘCIE W LEWO
7150 FOR K=20533 TO 22325 STEP 256
7160 LET L=2*PEEK K: IF L>255 THEN  LET L=L-256
7165 POKE K,L
7170 NEXT K: RANDOMIZE USR 65273
7180 RETURN
7299 REM  WCZYTANIE DANYCH Z TAŚMY
7300 INPUT "PODAJ NAZWĘ ZBIORU ";N$: IF N$="" THEN  RETURN
7320 LOAD N$CODE 64472,768
7340 PRINT #0; "ZATRZYMAJ TAŚME   - NACIŚNIJ    DOWOLNY KLAW
ISZ "
7350 PAUSE 0: RETURN
7500 REM  ZAPIS DANYCH NA TAŚMĘ
7510 INPUT "PODAJ NAZWĘ ZBIORU";N$: IF N$="" THEN  RETURN
7520 SAVE N$CODE 64472,768
7550 GO TO 7340
```

```
 !"#$%&'()*+,-./0123456789:;<=>?
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑_
£abcdefghijklmnopqrstuvwxyz{|}~©
£abcdefghijklmnopqrstuvwxyz{|}~©
```

```
1              WYSWIETLANIE...W
2              POPRAWIANIE..1-8
3              OBROT........L.R
4              ODBICIE......O.P
5              KODOWANIE......K
6              SUMA...........S
7              ZAPIS..........Z
8              PRZESUNIECIE.X.M
 12345678
```



Wygląd planszy programu.

```
R$ (K+31): NEXT K
 160 RANDOMIZE USR 65307
 170 PRINT #0;"CZY WCZYTAC DANE
Z TASMY ? T-N "
 180 GO SUB 6010
 190 IF S$="T" THEN GO SUB 7300
 195 CLS
 200 GO SUB 5000: GO SUB 5100: R
EM WYDRUK PLANSZY
 210 REM  PETLA GLOWNA
 220 GO SUB 6000
 230 IF S$>"0" AND S$<"9" THEN G
O SUB 6400
 240 IF S$="W" THEN GO SUB 6500
 245 IF S$="K" THEN GO SUB 6600
 250 IF S$="L" THEN GO SUB 6700
 255 IF S$="X" THEN GO SUB 7150
 260 IF S$="R" THEN GO SUB 6800
 265 IF S$="P" THEN GO SUB 6830
 270 IF S$="O" THEN GO SUB 6790
 275 IF S$="S" THEN GO SUB 7000
 290 IF S$="Z" THEN GO SUB 7500
```

Fragment listingu drukowanego pogrubionymi literami.

PRZERWANIE NMI —W ZX SPECTRUM—

Ze względu na konieczność programowania pamięci EPROM prezentowany układ polecany jest do wykonania bardziej zaawansowanym w tych pracach Czytelnikom.

Kilka błędów w oprogramowaniu ROM-u ZX SPECTRUM może sprawić czasem trochę kłopotu. Chyba najważniejszym z błędów jest zła zawartość komórki o adresie 112, w której zamieniono kod E9 (dziesiętnie 233) kodem E1 (dziesiętnie 225). Uniemożliwia to wykorzystanie przerwania niemaskowalnego NMI, ponieważ zamiast rozkazu JP (HL) znajduje się rozkaz POP HL. Przerwanie NMI przyjmowane jest przez mikroprocesor w każdym momencie, a program jego obsługi może być umieszczony w określonym przez programistę miejscu pamięci RAM. Daje to duże możliwości np. skopiowanie ekranu w dowolnym momencie gry itp.

Rozwiązanie układowe jest rozwinięciem wcześniej prezentowanego przy opisie wykorzystania wolnej części pamięci ROM w jednym z poprzednich numerów „BAJTKA". Zastosowana w rozwiązaniu pamięć EPROM 2716 umożliwi wprowadzenie poprawionej zawartości komórki o adresie 112, wykorzystanie 1166 bajtów wolnego obszaru ROM-u oraz ewentualne przeprogramowania generatora znaków.

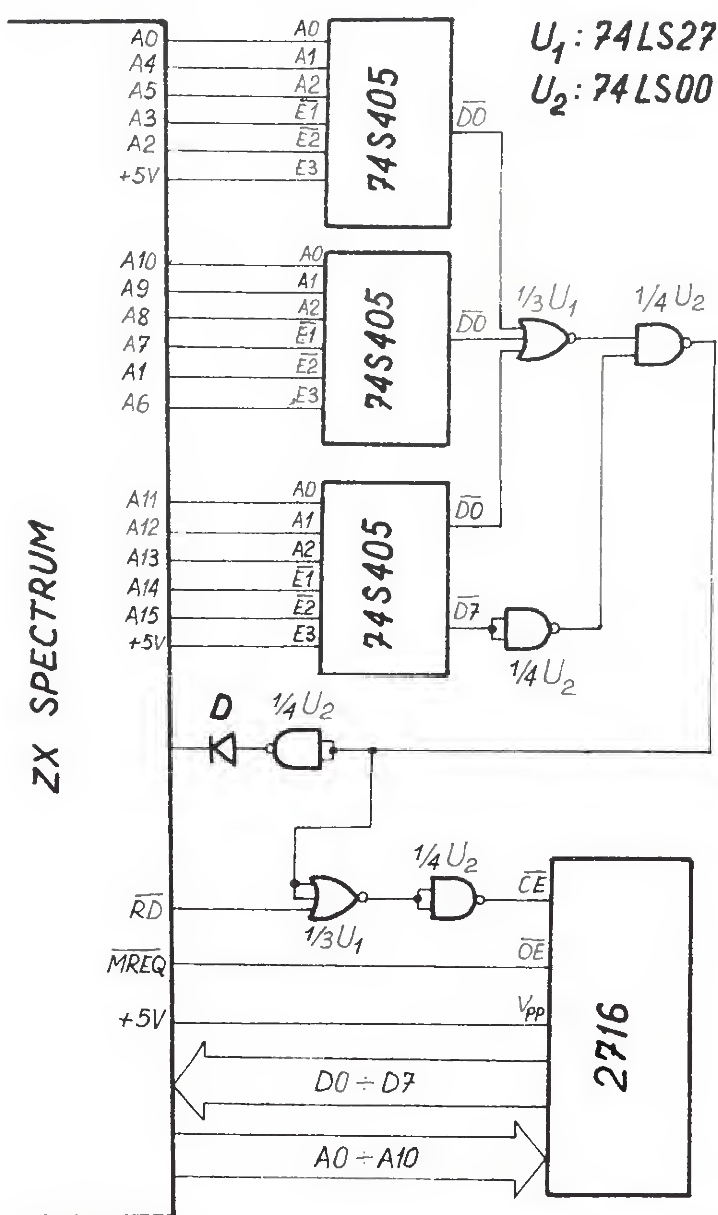
Istota rozwiązania przedstawionego na rys. 1 polega na zdekodowaniu adresu 112 w momencie odwoływania się systemu do komórki ROM-u o tym adresie, wyłączeniu ROM-u sygnałem ROMCS na moment pobierania rozkazu z tej komórki i uaktywnieniu EPROM-u sygnałem CE, w którego komórce o adresie bezwzględnym 112 wstawiono kod E9 [instrukcja JP (HL)]. Dodatkowo układ dekodera umożliwia wyłączenie ROM-u i zastąpienie go EPROM-em w momencie, gdy na magistrali adresowej pojawi się adres większy od 14445.

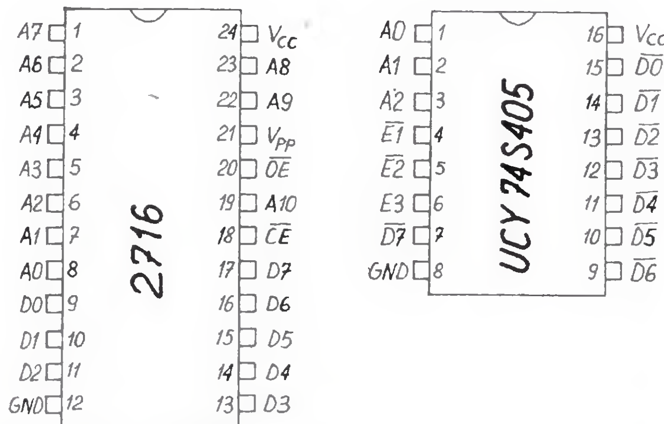
Wykorzystując NMI należy pamiętać, że w chwili podania sygnału przerwania nastąpi skok do podprogramu obsługi NMI, którego adres należy umieścić w zmiennych systemowych o adresach 23728 i 23729. Podprogram obsługi NMI musi kończyć się rozkazami:

```
POP HL
POP AF
RETN
```

ab umożliwić bezkolizyjny powrót do wcześniej wykonywanego programu. Sposób zaprogramowania EPROM-u przedstawia tabela 1. Ponieważ EPROM umieszczony jest w przestrzeni adresowej od 14336 do 16383, dlatego część ROM-u od adresu 14336 do 14445 włącznie należy przekopiować do EPROM-u. W obszarze tym znajduje się część interpretera języka BASIC. W komórce o adresie bezwzględnym EPROM-u 112 należy wpisać wartość 233. Obszar od adresu 113 do 1279 można wypełnić własnym oprogramowaniem, natomiast generator znaków rozpoczynający się w ROM-ie od adresu 15616 do 16383 można przekopiować lub zaprogramować na nowo. Układ wg schematu ideowego należy zmontować na płytce z obwodem drukowanym i za pomocą złącza krawędziowego połączyć z mikrokomputerem.

Konrad Fedyna
Zygmunt Wereszczyński





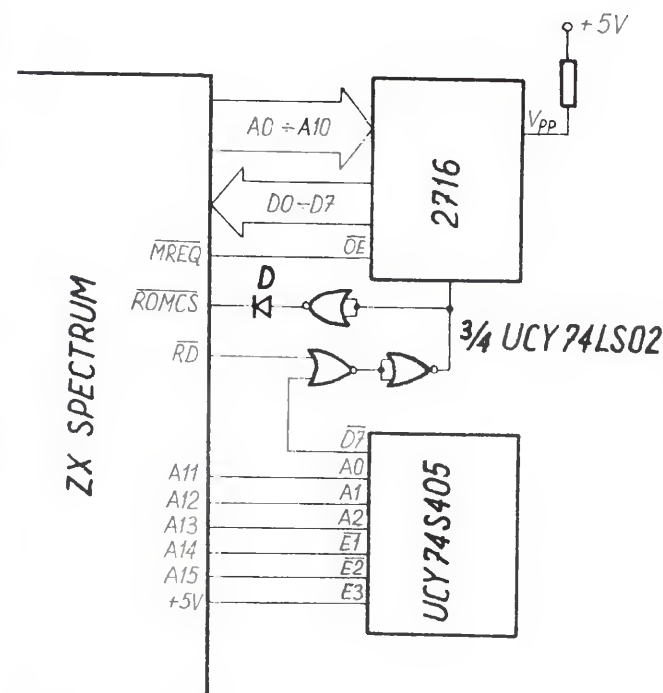
WYKORZYSTANIE WOLNEJ PAMIĘCI W ROM ZX SPECTRUM

Wykonanie tego układu wymaga znajomości podstaw techniki mikroprocesorowej i programowania w kodzie maszynowym, więc polecane jest bardziej zaawansowanym w takich pracach czytelnikom.

Mikrokomputer ZX SPECTRUM wyposażony jest w 16 KB pamięci ROM, w której umieszczono między innymi interpreter języka BASIC i generator znaków. Oprogramowanie znajdujące się w ROM-ie nie zajmuje jednak całej przestrzeni adresowej tzn. 16384 bajtów, lecz tylko jej część. Wolna część pamięci ROM to 1170 bajtów począwszy od adresu 14446 do 15615. Interpreter języka BASIC kończy się na komórce o adresie 14445, a od adresu 15616 do 16383 znajduje się generator znaków. Możliwość wykorzystania wolnych 1170 bajtów pamięci stałej daje wiele ciekawych możliwości np. można umieścić tam program obsługi drukarki itp.

Przykładowe rozwiązanie tego problemu przedstawiono na rys. 1. W konstrukcji wykorzystano pamięć EPROM typu 2716, która umieszczona jest w przestrzeni adresowej ROM-u od adresu 14336 do 16383. Pokrywa więc ostatnie 2 KB standardowego ROM-u SPECTRUM. Po zdekodowaniu adresu większego od 14335 przez układ dekodera zbudowanego przy pomocy układu UCY 74S405 produkcji krajowej, wyłączany jest ROM ZX SPECTRUM za pomocą sygnału ROMCS i jednocześnie uaktywniany jest EPROM przy pomocy sygnału CE. Funkcję wyłączonego ROM-u przejmuje wtedy zaprogramowany przez nas EPROM. Z uwagi na to, że interpreter języka BASIC kończy się na adresie 14445, więc część ROM-u od adresu 14336 do 14445 należy przekopiować do EPROM-u. Zajmie to obszar w wartościach bezwzględnych EPROM-u od 0 do 109. Następne 1170 bajtów tej pamięci można wykorzystać do umieszczenia własnego oprogramowania. Przy okazji mamy szansę zmiany kształtu znaków ZX SPECTRUM przez zaprogramowanie nowego generatora znaków. Standardowo zajmuje on obszar 768 bajtów od adresu 15616. Gdy zmiany generatora znaków nie przewidujemy, należy przekopiować obszar ROM-u od adresu 15616 do 16383. Tak przygotowany EPROM należy umieścić w podstawce na płytce z obwodem drukowanym wykonanym wg schematu ideowego. Całość łączymy z mikrokomputerem za pomocą złącza krawędziowego.

Konrad Fedyna
Zygmunt Wereszczyński



PRZEPRASZAMY

Program „Elektron w studni" z numeru 4 „Bajtka" ukazał się przez pomyłkę bez podpisu, za co serdecznie przepraszamy autora, Michała Małachowskiego i wszystkich czytelników. Artykuł był zresztą szczególnie pechowy — niemiły figiel spłatała nam drukarka wstawiając w listingu niepotrzebne spacje. Na szczęście nie ma to większego wpływu na działanie programu, z wyjątkiem linii 10, której prawidłową postać przedstawiamy poniżej.

```
 10 PRINT "WODOR - MODEL STUDNI
 POTENCJALOWseria
   seria:  LYMANA             BAL
MERA PASHENA     ↑
   ↑   ↑0.7"´TAB 6;"↑
 ↑     2.6"´" 12.7                v
1.9 v"´TAB 10;"↑"´TAB 4;12.1´´TA
B 7;10.2´TAB 17;"emisja swiatla:
                 seria PASHENA"´
TAB 17;a$´"  FUNKTY:          0.66
 eV"´´TAB 17;"seria BALMERA"´"
ENERGIA:          ";a$´TAB 19;"1.89
,2.55"´TAB 13;"*"´" *********
   seria LYMANA       *  ";m;"  *
 12.7 ";a$´" *********  /eV/  10
.2,12.1,12.7"
```

WARSAW BASIC

Nieczęsto się zdarza, żeby program trafiający w ręce użytkownika spełniał wszystkie pokładane w nim nadzieje, wszystkie wymagania. W przypadku poszczególnych dialektów BASIC dla Commodore 64 jest to sprawa szczególnie ważna, gdyż jak wiadomo, zaimplementowany na C 64 BASIC V2.0 daleki jest od doskonałości. Stąd też na rynku pojawiło się wiele rozszerzeń tego interpretera — SIMON'S BASIC, METABASIC, ULTRABASIC, OMIKRON BASIC i wiele innych. Ich twórcy starali się jednakże o rozszerzenie tylko niektórych możliwości komputera — np. graficznych czy muzycznych — dodając jakby „przy okazji" kilka funkcji i instrukcji poprawiających oryginalny edytor C-64. Jest wszakże jeden interpreter, którego autorzy poszli w zupełnie innym kierunku tworząc wersję języka BASIC zamieniającego Commodore 64 w maszynę prawie profesjonalną — WARSAW BASIC.

O interpreterze tym krąży już sporo mitów i legend. Przy okazji warto zauważyć, że krąży również kilka wersji tego programu, które — co tu ukrywać — zostały skradzione jego twórcom na wcześniejszych etapach produkcji i co za tym idzie są niewykończone i obarczone błędami.

WARSAW BASIC nie jest następną próbą ułatwienia życia amatorom gier czy rozwiązaniem problemów z uzyskiwaniem dźwięku na C-64 — wprowadza on wiele takich elementów jakie dostępne są praktycznie wyłącznie na maszynach profesjonalnych, co z kolei zbliża użytkownika do „dorosłej" problematyki informatycznej, do języków stosowanych powszechnie jedynie na maszynach dużych.

Autorami Warsaw Basic-a są dwaj warszawscy matematycy Krzysztof Gajewski i Bogusław Radziszewski z Instytutu Podstawowych Problemów Techniki. Dzięki ich uprzejmości „BAJTEK" otrzymał do testowania najnowszą wersję — WARSAW BASIC 3.2, zapisaną w pamięci EPROM wraz z dyskietką na której zawarte były procedury uzupełniające. Oprócz tego otrzymaliśmy także dobrze wykonaną instrukcję obsługi, która w jasny, konkretny i rzeczowy sposób wprowadza Czytelnika w możliwości tego języka.

Na pierwszy rzut oka WARSAW BASIC przypomina SIMONS BASIC. Złudzenie mija jednak szybko, w kilka minut po uruchomieniu programu demonstracyjnego. SIMON'S BASIC ułożył 14-letni chłopciec, WARSAW BASIC opracowali natomiast zawodowi matematycy. Z tego powodu istnieją duże różnice zwłaszcza w kierunku rozszerzenia jak też i możliwościach obu tych interpreterów. Walory WARSAW BASIC-a czynią go wprost idealnym językiem dla zastosowań matematycznych i edukacyjnych, z wyraźnym ukierunkowaniem na te drugie.

PROCEDURY

Podstawowym argumentem przeciwników wszelkich odmian BASIC-a jest brak możliwości programowania strukturalnego i możliwości tworzenia procedur. WARSAW BASIC daje programiście te możliwości i choćby z tego jedynie powodu plasuje się w rzędzie najbardziej nowoczesnych języków programowania wysokiego poziomu zaimplementowanych na komputerach ośmio-bitowych. Zachowuje przy tym wszystkie cechy BASIC-a, od którego — czy się nam to podoba, czy też nie — rozpoczyna edukację większa część programistów.

Procedury stanowią również rozwiązanie dla tych wszystkich, którzy potrzebują więcej pamięci do swoich zastosowań, o wiele więcej niż C-64 jest w stanie dać. Zamiast umieszczać w programie wszystkie niezbędne do jego wykonania bloki funkcjonalne, możemy stworzyć jedynie program, który w danej chwili gdy blok ten ma być wykonany będzie go wczytywał do pamięci z dyskietki czy kasety. O zaletach takiego rozwiązania w zasadzie nie trzeba mówić więcej — program korzystający z procedur zajmuje znacznie mniej pamięci, jest bardziej czytelny i przejrzysty, szybszy w działaniu. Ponieważ procedur nam potrzebnych możemy stworzyć ile dusza zapragnie, nie jesteśmy limitowani pojemnością pamięci. Zbiór procedur bibliotecznych na dysku A możemy w każdej chwili zastąpić dyskiem B. Jeśli poszukiwanej przez nas procedury nie ma na włożonym do stacji dysku, to WARSAW BASIC nie powoduje przerwania wykonywania programu i powrotu do trybu ekranowego; zamiast tego wyświetlany jest komunikat o braku takiej poszukiwanej procedury i żądaniu zmiany dyskietki.

Każdy program może więc być bardzo krótki, co z kolei stwarza możliwości jego rozbudowy jakich nie mieliśmy do tej pory w żadnym interpreterze BASIC-a (wszystkie bloki funkcjonalne programu musiały być umieszczone w nim na stałe). W ten sposób zdjęto z programistów jeden z największych i najbardziej kłopotliwych ciężarów występujących zwykle w pracy z mikrokomputerami — ograniczenie pojemności pamięci.

Nie koniec na tym. Procedury można nakładkować, zagnieżdżać, integrować w pakiety. Autorzy pomyśleli tu chyba o wszystkim, gdyż istnieje także możliwość tworzenia bloku wspólnego, deklarowania zmiennych lokalnych i parametrów formalnych. W samym programie procedurę można wywołać zarówno nazwą jak też i numerem linii; podczas jego pracy możliwe jest łączenie poszczególnych procedur np. wczytywanych z dyskietki.

Malkontenci mogliby zarzucić, że współpraca C-64 ze stacją dysków jest bardzo powolna. I ten zarzut WARSAW BASIC odpiera łatwo — jest on wyposażony w specjalną procedurę przyspieszającą współpracę z dyskiem czterokrotnie.

POLSKIE LITERY

Tekst, który zamierzamy wyświetlić na ekranie bądź wydrukować na drukarce może być złożony z dowolnych, zaprogramowanych przez użytkownika znaków, które to procedury mogą być rzecz jasna, zapisane na dyskietce lub taśmie. Oczywiście istnieje możliwość pracy z polskim alfabetem jak też i jakimkolwiek innym — odpowiedni zestaw może być w dolnym miejscu programu zmieniony na inny. Koniec z komunikatami typu BLAD SKLADNI czy ZADANIE KATA NA LACE. Koniec ze sztuczkami, z dziesiątkami POKE i innymi procedurami zastępczymi umożliwiającymi tworzenie własnych zestawów. Znaki raz zapisane mogą być wielokrotnie zmieniane w programie, podobnie jak i krój czcionki. Wszystkie zaprogramowane wcześniej zestawy czy alfabety mogą być wyprowadzone zarówno na ekran jak i na drukarki typu MPS 801—803, STAR, GEMINI i inne (bez względu na to czy są podłączone do portu szeregowego czy też do RS-232).

Zaleta ta może — moim zdaniem — zaspokoić wymagania nawet najbardziej wybrednych filologów. Za pomocą poniżej opisanych procedur można tworzyć bardzo rozbudowane programy językowe służące zarówno do ćwiczenia pisowni jak też gramatyki czy składni. Wszystko to wraz z alfabetem właściwym dla danego języka zamienia C-64 w bardzo pomocne i przydatne w szkole urządzenia. Stwarza to także olbrzymie możliwości do przygotowywania materiałów szkolnych pisanych choćby i po hebrajsku czy pismem klinowym...

GRAFIKA WYSOKIEJ ROZDZIELCZOŚCI (HIRES)

Coś dla amatorów grafiki komputerowej ale również dla nauczycieli przedmiotów ścisłych. WARSAW BASIC umożliwia przenoszenie do wybranego obszaru pamięci danego obrazu, jego zapis i wczytanie z dyskietki czy kasety, przesłanie na drukarkę, tworzenie obrazów dwu i trójwymiarowych (!), mieszanie tekstu i grafiki na ekranie.

Trzy omówione wyżej możliwości tego interpretera kwalifikują go do grona najlepszych i stwarzają wszystkim posiadaczom Commodore 64 zupełnie nowe horyzonty. Nie koniec na tym, autorzy pomyśleli również o innych wadach oryginalnego interpretera i wprowadzili kolejne ulepszenia.

Edytor wzbogacony został o bardzo przydatne nowe rozkazy i instrukcje takie jak AUTO, RENUM, DEL, FIND, MERGE, co znacznie ułatwi pracę podczas wpisywania czy edycji programu.

Matematycy otrzymali do dyspozycji osiem nowych funkcji (EVAL, EXAM, FRAC, HEEK, MAX, MIN, ODD, ROUND) oraz możliwości programowania strukturalnego (WBIF, WBELSE, WBEND, WHILE...DO, REPEAT...UNTIL). Warsaw Basic dysponuje także „procesorem tekstu w pigułce" (CMD*) oraz prostymi rozkazami i funkcjami umożliwiającymi zarządzanie zbiorami typu REL (zbiory o dostępie bezpośrednim) w znacznie bardziej przystępny sposób — służą do tego instrukcje CREATE, PRINT*, INPUT*, CLOSE*, OPEN*. Uciążliwe w użyciu komendy współpracy ze stacją dysków zostały zastąpione odpowiednimi symbolami (podobnie jak w DOS WEDGE (DOS 5.1) co znakomicie tą współpracę upraszcza. Wprowadzona została rozszerzona wersja instrukcji TRACE (może ona wskazywać zarówno poszczególne linie wykonywanie aktualnie przez komputer jak też i za pomocą kursora wskazywać poszczególne rozkazy w tej linii które są w danej chwili wykonywane). Pułapki bez wyjścia czyli napotkane błędy pozwala nam wyłapać bez przerywania programu zaimplementowane także w WARSAW BASIC ON ERROR... GOTO oraz RESUME.

O innych cechach WARSAW BASIC-a można się przekonać samemu studiując uważnie listę dodatkowych funkcji, instrukcji i rozkazów, którą zamieszczam poniżej. Dodam jeszcze, że WARSAW BASIC akceptuje wszystkie programy napisane dla Commodore 64 w wersji BASIC 2.0.

AXIS — służy do deklarowania parametrów rzutu z przestrzeni trójwymiarowej na płaszczyznę ekranu (wysokości rozdzielczości).

AUTO — umożliwia automatyczną numerację linii podczas wpisywania programu.

BEEP — wywołuje sygnał dźwiękowy o regulowanej głośności, wysokości i czasie.

CALL — pozwala na wywołanie i wczytanie do pamięci podprogramu o podanej nazwie.

CHAIN — umożliwia wczytanie programu z pamięci zewnętrznej i jego wykonanie; w programie tym można wykorzystywać zmienne deklarowane w programie wykonywanym uprzednio (nie zostają one automatycznie skasowane jak w wypadku LOAD).

CLOSE* — zamyka dostęp do otwartego zbioru relatywnego (REL).

CMD* — jest to odpowiednik HARDCOPY umożliwiającej przeniesienie zawartości ekranu na drukarkę, z tym, że w wypadku WARSAW BASIC możliwe jest podanie takich parametrów jak lewy margines, liczba wierszy liczba znaków w wierszu itp.

CMD PRINT — CMD polega tu na wciśnięciu firmowego klawisza Commodore oznaczonego jako Commodore LOGO Key. Wciskając np. ten klawisz i literę A otrzymamy literę „ą" itd.

COLOUR — umożliwia zmianę kolorów ramki, tła i tekstu.

COMMON — pozwala na utworzenie wspólnego bloku, do którego przepisywane są wszys-

tkie uprzednio zadeklarowane tablice i zmienne. COMMON OFF likwiduje taki blok. Dopuszczalne jest kilkakrotne stosowanie COMMON w danym programie.

CREATE — umożliwia utworzenie na dyskietce zbioru danych typu relatywnego (REL) o dostępie bezpośrednim.

DEF USR — deklaruje adres początkowy funkcji zapisanej w języku wewnętrznym.

DEF EXTERN — deklaruje adres początkowy procedury zapisanej w języku wewnętrznym.

DEL — umożliwia kasowanie pojedyńczych linii jak też całych fragmentów programu.

DISPOSE — umożliwia skasowanie informacji zawartej na stosie mikroprocesora o ostatnio otwartej pętli FOR... NEXT (DISPOSE NEXT), podprogramie (DISPOSE RETURN) lub ustawia wskaźnik stosu na stan jak zaraz po inicjacji (DISPOSE CLR).

DOT — umożliwia włączenie (DOT) bądź wyłączenie (DOT OFF) danego punktu na ekranie wysokiej rozdzielczości (HIRES).

DOS — daje użytkownikowi możliwość współpracy ze stacją dysków za pomocą poleceń symbolicznych np. zamiast LOAD „PROGRAM",8 i RUN możliwe jest użycie formy ↑PROGRAM.

DRAW umożliwia tworzenie grafiki wysokiej rozdzielczości rysowanie linii.

EVAL — pozwala na zamianę wyrażenia tekstowego na wyrażenie arytmetyczne, innymi słowy możliwe jest (w dużym uproszczeniu) wpisywanie żądanych funkcji np. poprzez INPUT.

FIND — pozwala na wyszukanie określonego ciągu tekstu czy stałych numerycznych i wyświetlenie na ekranie numerów linii gdzie się one znajdują.

FRAC (X) — przyjmuje wartość równą części ułamkowej wyrażenia, które jest jej argumentem.

GOTO. — umożliwia skok do początku linii w którym znajduje się ten rozkaz. Zamiast pisać 20 PRINT A: GOTO 20 wystarczy wpisać 20 PRINT A:GOTO..

HEEK (X) — pełni tę samą funkcję co PEEK, czyli podaje zawartość danej komórki pamięci. Różnica polega na tym, że tam gdzie PEEK podaje zawartość ROM, HEEK poda nam zawartość RAM leżącej „pod" ROM.

HICOLOUR — określa kolor ramki, tła i tekstu na ekranie wysokiej rozdzielczości.

HILOAD — pozwala na wczytanie do pamięci zapisanego uprzednio obrazu wysokiej rozdzielczości.

HIMEM — określa obszar pamięci w którym będziemy tworzyć obraz wysokiej rozdzielczości.

HIPRINT — umożliwia mieszanie na ekranie wysokiej rozdzielczości grafiki oraz tekstu.

HIRES — pozwala na wyświetlenie obrazu wysokiej rozdzielczości na ekranie monitora czy odbiornika telewizyjnego.

HISAVE — umożliwia zapisanie na dysku/kasecie obrazu wysokiej rozdzielczości.

IF..THEN..ELSE — pełni tą samą funkcję co zwykłe IF THEN z tym, że dodano tu instrukcję ELSE oznaczającą „w przeciwnym wypadku wykonaj..."

INPUT* — pozwala na wczytanie danego rekordu ze zbioru typu relatywnego REL (o dostępie bezpośrednim).

KILL — umożliwia powrót do standardowego interpretera Commodore 64.

LINE — rysuje linię na ekranie wysokiej rozdzielczości.

LIST — w standardowym interpreterze, wykonanie LIST powoduje zawsze powrót do trybu pracy ekranowego (działanie programu w którym LIST wystąpiła zostaje przerwane). Wada ta została usunięta i LIST może być używany w programie, nie powodując jednocześnie wspomnianego wyżej powrotu do trybu ekranowego.

LIST. — podobnie jak w wypadku GOTO, powoduje wyświetlenie na ekranie linii w którym LIST. występuje.

MAX — przyjmuje wartość równą największej wartości spośród wyrażeń będących argumentami tej funkcji.

MERGE — pozwala na dołączenie do programu znajdującego się w pamięci komputera drugiego programu zapisanego na dysku czy taśmie.

MERGE PROC — umożliwia tworzenie zestawów (pakietów) procedur.

MEM — wyświetla na ekranie informację o aktualnej konfiguracji pamięci, liczbę bajtów zajętych przez tekst programu, poszczególne typy zmiennych, tablice itp.

MIN — tak jak MAX — tyle, że przypisywana jest wartość najmniejsza.

MOVE — umożliwia przeniesienie wybranych obszarów pamięci w inne jej miejsce, dotyczy to także grafiki wysokiej rozdzielczości.

NEW — oprócz zwykłej swojej funkcji kasuje również wspólny blok zmiennych. NEW PROC umożliwia także kasowanie wczytanych do pamięci podprogramów.

ODD — przyjmuje wartość — 1 gdy wyrażenie ma wartość nieparzystą.

ON ERROR GOTO — umożliwia obsługę błędu, który wystąpił bez przerywania wykonywania programu.

OPEN* — otwiera dostęp do zbioru relatywnego typu REL (o dostępie bezpośrednim).

PRINT@ — odpowiednik PRINT AT czyli pozwala na wyświetlenie danego tekstu w ściśle określonym miejscu ekranu.

PRINT USING — umożliwia formatowanie zapisu np. cyfr w pewnej z góry ustalonej formie na ekranie.

PRINT* — pozwala na zapisanie rekordu do zbioru o dostępie bezpośrednim (REL).

PROCEDURE — określa procedurę zdefiniowaną przez użytkownika. Procedura taka musi się kończyć PROC END.

PROC — umożliwia nakładkowanie, redagowanie i testowanie procedur zdefiniowanych uprzednio przez użytkownika.

REM — instrukcja REM może być zastąpiona w WARSAW BASIC za pomocą apostrofu.

RENUM — umożliwia przenumerowanie linii programu w żądanej przez użytkownika formie. Przenumerywowaniu poddawane są także numery linii występujące po GOSUB, GOTO, RUN itp.

RESTORE — pozwala na ponowne odczytanie danych zawartych w DATA wraz ze ścisłym określeniem miejsca skąd odczytywanie to będzie wykonywane.

RESUME — występuje jako zakończenie procedury obsługi błędu i pozwala na określenie linii od której ma zostać wznowione wykonywanie programu.

REVERS — umożliwia utworzenie negatywu danego obrazu wysokiej rozdzielczości.

ROUND — umożliwia zaokrąglenie danej liczby do liczby znaków określonych jednym z parametrów tej funkcji.

RUN. — pozwala na uruchomienie danej procedury (RUN PROC), uruchomienie programu od ostatnio wykonywanej linii lub linii gdzie RUN. się znajduje.

SDOT — umożliwia odzwierciedlenie na płaszczyźnie ekranu (w trybie wysokiej rozdzielczości) punktu który jest rzutem z przestrzeni trójwymiarowej.

SDRAW — tak samo jak DRAW z tym, że odnosi się do przestrzeni trójwymiarowej.

SEC — umożliwia zatrzymanie programu (pauza) na określoną liczbę sekund.

SLEEP — wyłącza ekran przyspieszając w ten sposób wykonywanie programów o ok. 8%, zmieniany jest też system przerwań.

SLINE — tak samo jak LINE z tym, że odnosi się do przestrzeni trójwymiarowej.

SWAP — pozwala na zamianę między sobą dwóch obszarów pamięci.

SYS — za pomocą określonych w instrukcji obsługi adresów można m.in. włączać i wyłączać przyspieszanie wczytywania programów z dysku, oraz inicjalizację klawiszy funkcyjnych (klawiszom tym przypisano 12 różnych funkcji).

TRACE — śledzenie wykonywanego programu.

WBIF — umożliwiają nam programowanie strukturalne.
WBELSE
WBEND

Możliwości oferowanych nam przez WARSAW BASIC jest oczywiście znacznie więcej, niestety ramy pisma nie pozwalają na ich dokładne przedstawienie.

Program ten powstawał i doskonalił się przez trzy lata. Trzy lata ciężkiej pracy dwóch zapaleńców, którzy postanowili udowodnić na przekór wszystkim, że pogardliwie traktowane przez „dorosłych" informatyków mikrokomputery mogą także przydać się niejednemu — wystarczy jedynie chcieć i umieć ułożyć do nich odpowiedni język.

WARSAW BASIC istnieje na razie tylko w wersji dla komputerów Commodore 64 i 128. Nic jednak nie stoi na przeszkodzie, by powstały wersje dla innych komputerów. Z pewnością warto, bo jest to jeśli nie najlepszy, to jeden z najlepszych języków, którymi dysponują komputery domowe i półprofesjonalne. Marzeniem autorów jest zainstalowanie WARSAW BASIC-a w komputerach szkolnych. Wszyscy w naszej redakcji zgodni są co do tego, że byłoby to znakomite rozwiązanie.

Klaudiusz Dybowski

POWIĘKSZENIE PAMIĘCI DLA COMMODORE 16

W roku 1984 firma Commodore przedstawiła trzy nowe modele mikrokomputerów: C-16, C-116 oraz Commodore PLUS 4. Modele te są poza niewielkimi różnicami (klawiatura, pamięć RAM, obudowa) identyczne. Mimo, że oferują znacznie lepszą wersję języka BASIC (V3.5), to pozostają jednak w cieniu Commodore 64. Jest to spowodowane m.in. małą ilością oprogramowania oraz (w wypadku C-16 i C-116) niewielką pojemnością pamięci RAM. Tę ostatnią wadę usunąć można poprzez dołączenie oddzielnego modułu albo — jak proponujemy wam w tym artykule — dokonując niewielkich przeróbek wewnątrz komputera.

Pamięć RAM w C-16 to dwa układy pamięci dynamicznej o pojemności 16K x 4 bit (układy U5 i U6 jak na rys.1). Do adresowania pamięci 16KB komputer wymaga tylko 14 linii adresowych (do A13). Układy pamięci są połączone z szyną adresową poprzez dwa multipleksery (układy U7 i U8). Dwie pozostałe linie adresowe (A14 i A15) nie są podłączone do szyny adresowej, ale do +5V aby miały określony stan (w tym przypadku wysoki).

Zwiększenie pamięci RAM uzyskuje się poprzez zastosowanie większych układów pamięci (64K x 4 bit), odłączenie obu nie wykorzystanych linii adresowych i połączenie ich poprzez układ U16 (zarządzający adresami) z szyną adresową. Jako układy pamięci można zastosować układy FUJITSU MB 81464-15, TMS 4464-15N2, TMN 41464P-12 czy też ich odpowiedniki.

Po wylutowaniu układów U5 i U6 (warto przedtem zajrzeć do „Bajtka" nr 2/86) wlutowujemy w te miejsca podstawki i osadzamy w nich nowe układy. Następnie należy odłączyć obie linie adresowe od zasilania +5V; w tym celu należy przeciąć skalpelem lub żyletką cienką ścieżkę poprowadzoną pod układem U7, łącząc nóżki 2 i 16. Może to pociągnąć za sobą konieczność wylutowania układu. W takim przypadku radzimy również wstawić w to miejsce podstawkę. W przypadku multipleksera U8 sprawa jest o wiele prostsza — należy tylko przeciąć po stronie druku połączenie nóżki 14 z szeroką ścieżką zasilania +5V.

Po przerwaniu połączeń należy dołączyć wolne linie adresowe do szyny adresowej. W tym celu należy za pomocą cienkich, izolowanych przewodów połączyć nóżkę 2 układu U8 z nóżką 7 układu U16 (linia A15) oraz nóżkę 14 układu U8 z nóżką 26 układu U16 (linia A14). Po zmontowaniu całości i włączeniu zasilania otrzymujemy na ekranie napis:

COMMODORE BASIC V3.5 60678 BYTES FREE.

W C-16 nie ma problemu z zarządzaniem większą ilością pamięci poza jednym wyjątkiem: w trybie graficznym pamięć BASIC znajduje się w innym obszarze niż w trybie tekstowym, co przy dłuższych programach graficznych może doprowadzić do pewnych komplikacji. Z tego powodu radzimy przełączać komputer w tryb graficzny bezpośrednio po włączeniu i powracać do trybu tekstowego rozkazem GRAPHIC 0 zamiast GRAPHIC CLR.

Witold Misztal
Sławomir Waszczuk

Rys. 1. Rysunek poglądowy wykonanej przeróbki

SUPER WTYK

Zapewne nie jeden z Was próbował przy pomocy zwykłego kabla do nagrywania połączyć poprzez gniazda do magnetofonu dwa mikrokomputery Meritum, w celu przegrania programu z jednego mikrokomputera na drugi.

Stosowanie zarówno kabla monofonicznego jak i stereofonicznego nie dawało pożądanych efektów, dlaczego? Otóż okazuje się, że gniazdo magnetofonowe w naszym mikrokomputerze podłączone jest w następujący sposób: na styku nr 1 podłączone wyjście, na styku nr 2 podłączona jest masa, a na styku nr 3 podłączone jest wejście. Łącząc teraz poprzez zwykły kabel nagrywający otrzymywaliśmy zawsze połączenie wejście-wejście, masa-masa, oraz wyjście-wyjście. Aby jednak można było przegrywać programy „komputer-komputer" należałoby doprowadzić do tego aby wejście było połączone z wyjściem i odwrotnie, wyjście było połączone z wejściem, a masy razem.

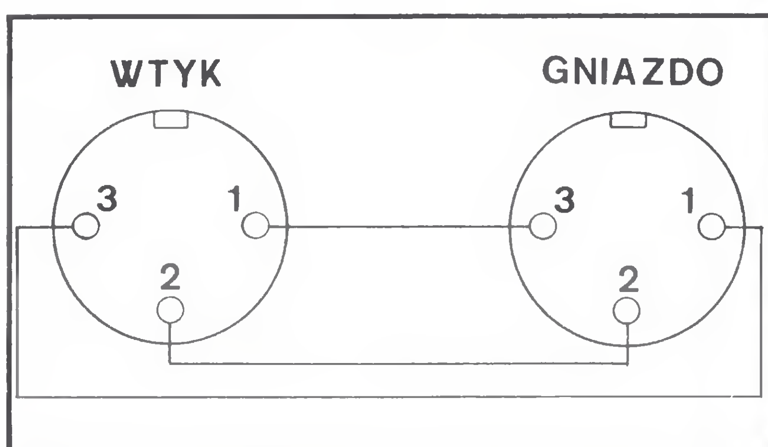
Niektórzy z Was przelutowaliby zapewne zwykły kabel nagrywający zmieniając tylko przewody miejscami, można i tak, ale wtedy potrzebne są dwa kable, jeden do nagrywania z magnetofonu, drugi do przegrywania programów „komputer-komputer", przy czym należy jeszcze pamiętać, który kabel do czego będzie służył. Proponuję Wam zbudowanie prostego złącza, które pozwoli na wykorzystanie jednego kabla nagrywającego. Do naszego „SUPER-WTYKU" potrzebna jest jedna wtyczka diodowa i jedno gniazdko diodowe. Gniazdko i wtyczkę łączymy w następujący sposób: styk nr 1 łączymy ze stykiem nr 3, styk nr 3 łączymy ze stykiem nr 1, styki nr 2 łączymy razem.

Teraz tylko zostaje nam wypróbowanie „SUPER-WTYKU", proponuję Wam program, który pozwoli na samoczynne wgranie się programu w przypadku pojawienia się sygnału w gniazdku magnetofonowym w mikrokomputerze.

```
10 A=INP(255)
20 IF A<>INP(255) THEN CLOAD
30 GOTO 20
```

Po uruchomieniu powyższego programu teraz już wystarczy tylko wysłać zlecenie CSAVE „nazwa" na drugim mikrokomputerze, aby na pierwszym mikrokomputerze program zaczął się wczytywać bez obawy że nie zdążymy w odpowiednim momencie wcisnąć ENTER. Program ten można też wykorzystać przy wczytywaniu programów z magnetofonu.

Bogdan A. Grzybowski
INFORMIK



Co piszczy pod klawiaturą?

(cz. 8)

TABELA ADRESÓW PROCEDUR SYSTEMOWYCH C.D.

Nr	Adres wektora	Adres rzeczywisty/opis 464	664	6128
94	BC1A	0B64	0B66	0B6A
		Oblicza adres rzeczywisty znaku na ekranie (kolumna, linia). Wej: H zawiera numer kolumny, a L numer linii. Wyj: HL zawiera adres rzeczywisty pamięci ekranu, B zawiera informację o ilości bitów na piksel w aktualnym trybie ekranu, AF są modyfikowane.		
95	BC1D	0BA9	0BAB	0BAF
		Oblicza adres rzeczywisty punktu na ekranie. Wej: DE zawiera współrzędną X punktu a HL współrzędną Y punktu licząc w stosunku do lewego dolnego rogu ekranu. Wyj: HL zawiera adres rzeczywisty punktu (piksela) w pamięci, B ilość bitów/piksel zmniejszoną o jeden, C zawiera maskę punktu. AF i DE są modyfikowane.		
96	BC20	0BF9	0C01	0C05
		Oblicza adres rzeczywisty dla bajta w prawo od adresu rzeczywistego bieżącego. Wej: HL zawiera adres bieżący. Wyj: HL zawiera nowy adres a AF są modyfikowane.		
97	BC23	0C05	0C0D	0C11
		Podobnie jak dla BC20, lecz dla bajta w lewo		
98	BC26	0C13	0C1B	0C1F
		Podobnie jak dla BC20, lecz dla odpowiedniego bajta w niższej linii.		
99	BC29	0C2D	0C35	0C39
		Podobnie jak dla BC20, lecz dla odpowiedniego bajta w wyższej linii.		
100	BC2C	0C86	0C8A	0C8E
		Maskuje numer atramentu w taki sposób, aby po dostarczeniu bajta reprezentującego punkty nastąpiło wyświetlenie punktów we właściwym kolorze. Wej: A zawiera numer atramentu. Wyj: A zawiera maskę a F jest modyfikowany.		
101	BC2F	0CA0	0CA3	0CA7
		Wykonuje proces odwrotny w stosunku do poprzedniego (BC2C). Wej: A zawiera maskę. Wyj: A zawiera numer atramentu a F jest modyfikowany.		
102	BC32	0CEC	0CEE	0CF2
		Ustala kolory atramentu. Wej: A zawiera numer atramentu, B zawiera pierwszy kolor a C drugi kolor. Wyj: AF, BC, DE i HL są modyfikowane.		
103	BC35	0D14	0D16	0D1A
		Odczytuje kolory atramentu. Wej: A zawiera numer atramentu. Wyj: B zawiera pierwszy kolor a C drugi kolor. AF, DE i HL są modyfikowane.		
104	BC38	0CF1	0CF3	0CF7
		Ustala kolory wyświetlania ramki (BORDER). Wej: B zawiera pierwszy kolor, C zawiera drugi kolor. Wyj: AF, BC, DE i HL są modyfikowane.		
105	BC3B	0D19	0D1B	0D1F
		Odczytuje kolory ramki (BORDER). Wej: nie ma. Wyj: B zawiera pierwszy kolor, C zawiera drugi kolor. AF, DE i HL są modyfikowane.		
106	BC3E	0CE4	0CE6	0CEA
		Ustala czasy migania kolorów ramki. Wej: H zawiera czas wyświetlania pierwszego a L drugiego koloru. Wyj: AF i HL są modyfikowane.		
107	BC41	0CE8	0CEA	0CEE
		Odczytuje czasy migania kolorów ramki. Wej: nie ma. Wyj: H zawiera czas wyświetlania pierwszego a L drugiego koloru. AF są modyfikowane.		
108	BC44	0DB3	0DB5	0DB9
		Wypełnianie prostokąta atramentem. Wej: A zawiera maskę odpowiadającą atramentowi, H numer kolumny lewej, D numer kolumny prawej, L numer górnej linii a E numer dolnej linii. Wyj: AF, BC, DE i HL są modyfikowane.		
109	BC47	0DB7	0DB9	0DBD
		Ustala ilość bajtów w pamięci ekranu dla wypełniającego prostokąt atramentu. Wej: A zawiera maskę odpowiadającą atramentowi, HL zawiera adres odpowiadający lewemu górnemu rogowi prostokąta, D zawiera ilość bajtów, E ilość linii. Wyj: AF, BC, DE i HL są modyfikowane.		
110	BC4A	0DDF	0DE1	0DE5
		Zamiana dwóch kolorów znaku. Wej: B zawiera maskę jednego koloru, C zawiera maskę drugiego koloru, H zawiera numer kolumny a L numer linii. Wyj: AF, BC, DE i HL są modyfikowane.		

Wojciech Ziółek

DAN DARE
(PILOT OF THE FUTURE)

Dan Dare to barwna i popularna postać z angielskich komiksów typu SF. Dan to rodzaj bohatera gotowego oddać życie za powodzenie swojej misji. Przedstawiane w programie przygody Dana są związane z pojawieniem się gróźb ze strony Mekona. Mekon to typowy czarny charakter choć jego skóra miała kolor raczej zielony.

Mekon zmienił kurs jednego z asteroidów krążących wokół Plutona na kolizyjny z Ziemią. Na asteroidzie założył swoją bazę i stamtąd wysyła żądania wobec Ziemian. Na zasadzie szantażu próbuje wymusić na Ziemianach objęcie przez siebie protektoratu nad Ziemią.

Ziemianie mają tylko dwa wyjścia z tej sytuacji:
— zgodzić się na żądania Mekona i oddać się w niewolę
— czekać na rozwój wypadków i mieć cień nadziei na ratunek.

W niecnych poczynaniach, Mekonowi pomaga jego gwardia przyboczna złożona z doborowych oddziałów Treens'ów.

Rząd Światowy podjął jednak decyzję o wysłaniu specjalnej misji do wysadzenia całego asteroidu wraz z bazą Mekona.

W skład załogi „Anastazji" („Anastazja" — statek typu „SCOUT" o napędzie fotonowym) weszły dwie osoby Digby i Dan Dare. Przy czym Dan ma za zadanie uruchomić system samozniszczenia asteroidu a Digby przygotowuje ucieczkę z zagrożonej strefy.

Na wyposażeniu Dan ma wysokoenergetyczny pistolet laserowy i profilaktyczną osłonę siłową o niewielkiej mocy. Broń i osłonę w trakcie misji można doładować.

Cel gry:

Po przylocie „Anastazją" na asteroid trzeba zejść do podziemi bazy. Baza ma pięć poziomów i na każdym jest część mechanizmu autodestruktora. Każdego poziomu bronią „Treens'i a ponadto roboty, „strzelające miny", zapadnie i inne niespodzianki. Każda część autodestruktora wygląda jak mała beczka a zabierzesz ją gdy wskoczysz na nią.

Wszystkie części autodestruktora trzeba przynosić do Centrum Kontroli asteroidu (pomieszczenie łatwo odnaleźć — „pięć kółek na dwóch słupach"). Każde przyniesienie części powoduje pracę części systemu.

Trzeba też dodać, że na odnalezienie i skompletowanie systemu autodestruktora masz niecałe dwie godziny czasu symulacji.

Każdorazowe odniesienie części powoduje też otworzenie drzwi następnego poziomu.

Windy (stawaj „w cieniu" i wciskaj klawisz zgodny z kierunkami strzałek) poruszają się w kierunkach wskazywanych przez strzałki.

Używaj broni bardzo oszczędnie gdyż ma ograniczoną moc.

„Minę" można zniszczyć, wskakując na nią.

Uważaj na strzelające z sufitu lasery.

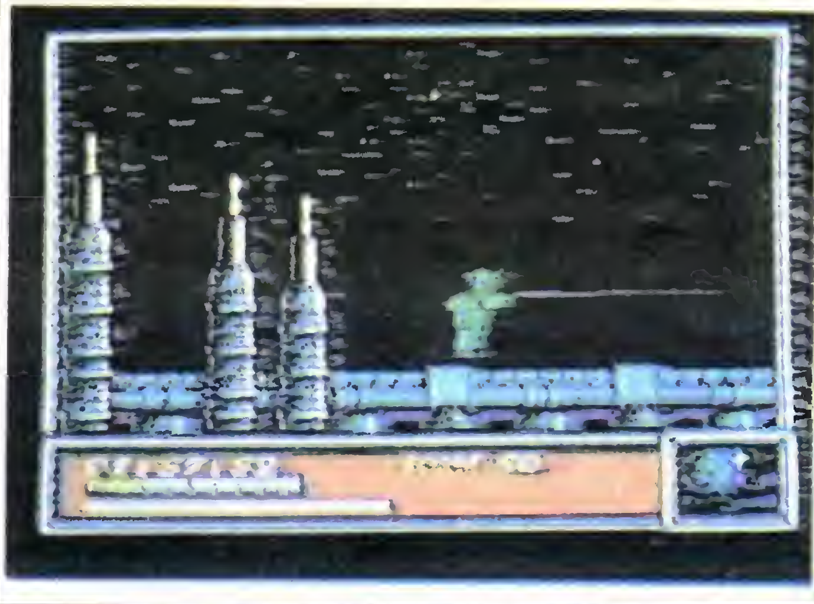
Gdy zabraknie ci energii idziesz odsiedzieć 10 min. za kratki po czym uciekasz i znów jesteś wolny.

Gdy uciekasz z „więzienia" nie obawiaj się nikogo i niczego gdyż bez względu na twoje poczynania energia przez pewien czas nie maleje.

„Powodzenia — powiedział przed odlotem prezydent Ziemi — losy ludzkości w twoich rękach Dan. Wierzymy w Ciebie, nie zawiedź nas. — Potem „Anastazja" przetransportowała nas na asteroid. Już nie ma odwrotu".

(M.1)

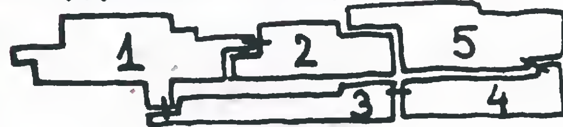
Firma — **VIRGIN GAHEG**
Autor — **The Gang of Five**
Komputer — **ZX Spectrum 48/+/128/+2/+3**



RYS: M.STAWICKI

DAN DARE

— PRZEJŚCIA MIĘDZY SEKTORAMI GRY



- D NAZIEMNE DZIAŁO
- ● ENERGIA
- AMUNICJA
- WINDA
- □ KLUCZ (SDS KEY)
- LASER ŚCIENNY
- ▶ DZIAŁKO ŚCIENNE
- WIĘZIENIE

MEKON

POZIOM 5

POZIOM 4

POZIOM 2

POZIOM 3

10 BAJTKOWA LISTA PRZEBOJÓW (7/87)

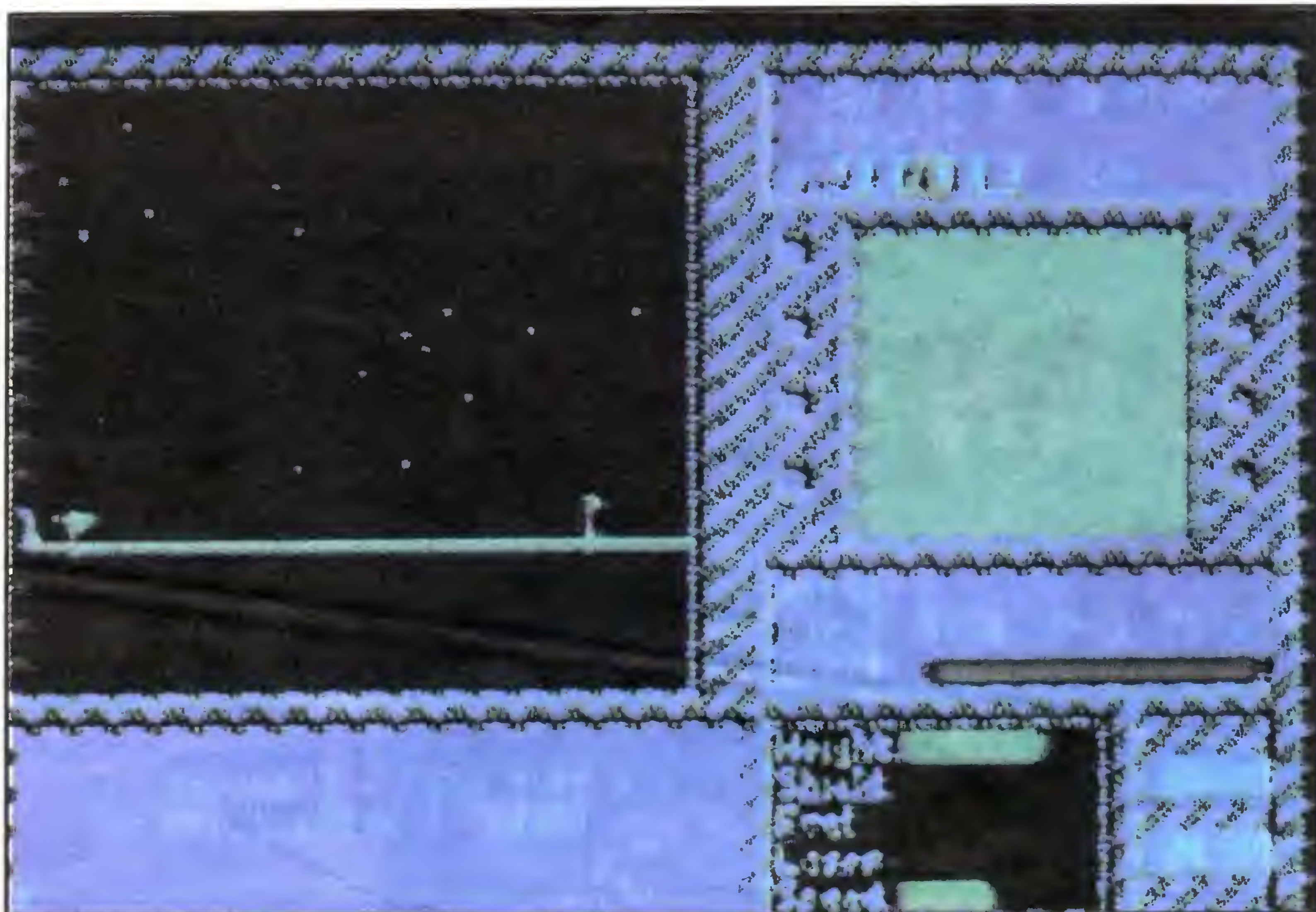
Ostatnio dostałem wiele listów od nowych czytelników Bajtka z prośbą o przypomnienie zasad głosowania na listę. Przypominam, każdy głosujący nadsyła własną, jego zdaniem najlepszą dziesiątkę gier na dowolny mikrokomputer.
Wszystkie propozycje zostają zsumowane i tworzą „złotą 10". Główną nagrodę otrzymuje osoba której propozycja jest zgodna z listą zamieszczoną w Bajtku (jak dotąd nikt nie trafił). Niezależnie zawsze spośród nadesłanych propozycji wybieramy Bajtkowego Króla i Królową Gier (Specjalna premia za opisy typowanych gier). Na siódme notowanie napłynęło 4.320 propozycji, głosowano na 153 tytuły gier.

			ATARI	AMSTRAD	COMMODORE	SPECTRUM
1	BROADSIDES	▲	×		×	
2	SILENT SERVICE	▼	×		×	×
3	W.A.R.	▲			×	×
4	WARHAWK	▲	×		×	
5	WORLD KARATE CHAMPIONS HIPS	▲	×		×	
6	BEACH HEAD II	▼	×	×	×	×
7	SPY Vs SPY II	▼	×	×	×	
8	DAN DARE	!				×
9	WINTER GAMES	!	×	×	×	×
10	BOULDER DASH	▼	×	×	×	×

Nagrody — zestawy programów komputerowych — ufundowane przez firmę Electronics Export z Londynu otrzymują: Piotr Banaszewski z Łodzi oraz Radosław Zyskowski również z Łodzi.

Sławek





TAU CETI

TAU CETI reprezentuje nowy typ gier komputerowych, w których nastąpiło połączenie wielu rodzajów gier tj. gier zręcznościowych (ARCADE), gier strategicznych (STRATEGIC) i gier symulacyjnych (SIMULATIONS). Połączenie to w efekcie dało grę, w którą może grać każdy. Inne tego typu gry to: ELITE (firmy Firebird), ACADEMY (firmy CRL) czyli druga część TAU CETI.

Jaki jest cel gry i jak w nią grać?

Nowoczesnym, małym kosmicznym pojazdem musisz wylądować na wielkiej planecie robotów. Gdy wylądujesz, ruszy za tobą pościg więc musisz go zgubić lub zniszczyć (co w gruncie rzeczy jest o wiele prostsze). Gdy pozbędziesz się towarzystwa zacznij wykonywać zadaną misję. Główną częścią twojej misji jest odnalezienie 40 połówek prętów paliwowych do reaktora. Z dwóch pasujących do siebie połówek możesz (odpowiednio dopasowując, zmieniając kolory) złożyć cały pręt. Wszystkie te czynności wykonujesz na ekranie za pomocą opcji TODS. Po skompletowaniu całego materiału paliwowego możesz (wsuwając odpowiednie pręty na ich miejsca) uruchomić reaktor. To cała misja. Prawda że proste?

Cały świat na zewnątrz możesz obserwować i odpowiednio interpretować pod warunkiem, że perfekcyjnie poznasz wszystkie możliwości swojego pokładowego komputera.

Rozejrzyj się trochę po swojej tablicy rozdzielczej (masz ją na ekranie). Widzisz sporo różnych wskaźników. Dzięki nim będziesz mógł skutecznie kontrolować komputer a co za tym idzie i pojazd.

Oto tablica i znaczenie wskaźników:

HEIGHT — wskaźnik wysokości (w stopach „foot")
SHIELD — „tarcza" czyli pole ochronne pojazdu
FUEL — wskaźnik paliwa (w galonach)
LASER — laser, twoja broń
SPEED — szybkościomierz (w milach na godzinę)

Efektywna zdolność lasera jest ograniczona co widać na wskaźniku „FL" (nie używaj go zbyt długo, łatwo jest go przegrzać).

Masz też w polu widzenia wskaźnik podczerwieni (Noktowizor). Gdy masz włączone reflektory podczerwone twoja widoczność ulegnie poprawie.

U góry ekranu widzisz kompas ze swoim aktualnym kierunkiem ruchu, dalej zegar z aktualnym czasem gry (do skończenia jej potrzeba co najmniej 6 godzin zegarowych). Masz do spenetrowania 30 miast, a to w którym aktualnie jesteś, masz napisane na ekranie. Widzisz także informacje o aktualnym stanie twojego pojazdu (STATUS).

W lewym, dolnym rogu ekranu masz mały terminal do porozumiewania się ze swoim pokładowym komputerem. Możesz wydawać mu pewne rozkazy:

HELP — pomoc, przypomnienie rozkazów
SIGHTS ON — włącz reflektory podczerwone
SIGHTS OFF — wyłącz reflektory podczerwone
LOOK — przegląd aktualnej sytuacji
LAUNCH — załadowanie katapulty

Można zadawać pytania:

MAP — mapa, widok terenu
EQUIP — sprzęt, wykaz sprzętu, który posiadasz
STATUS — aktualny stan twego pojazdu

Możesz dzięki komputerowi wykonywać pewne operacje:

REACTOR — reaktor
RODS — elementy reaktora

Przydatne wskazówki do gry:

- mogą zaatakować cię wrogowie, niszcz ich laserem lub MISSIL'em.
- po opuszczeniu miasta wrogowie pojawią się z powrotem
- nie strzelaj w „drzwi", gdyż nie wpuszczą cię do środka
- paliwo możesz tankować w centrach wojskowych lub cywilnych
- w centrach możesz znaleźć też inne przydatne przedmioty.

(M.1)

Firma — **CRL**
Autor — **PETE COOKE**
Komputer — **ZX Spectrum 48k(+)128(+2)+3. Commodore 64/128, Amstrad / Schneider 464/6128**



CRITICAL MASS

...**„*Widząc wrogi ornitopter,*** *Paul Atryda mocniej zacisnął dłonie na sterach. Chwila skupienia, strzał i wrogi pojazd rozbił się o skały pustynnej planety Arrakis. To znowu sardaukarzy w służbie Harkonnenów — pomyślał Paul. Baron Vladimir Harkonnen bardzo łatwo pozbył się swych doskonałych oddziałów. Z zamyślenia wyrwał Paula nagły wstrząs ornitoptera. Pojazd rozbił się o szczątki przetwórni przyprawowej. Chroniąca Paula tarcza uratowała go przed śmiercią. Jedynym wyjściem był powrót do filtranamiotu po drugi ornitopter. Młody Atryda wiedział, że znajduje się na terenach zajętych przez czerwie pustynne. Te ogromne robaki (rdzenni mieszkańcy Arrakis nazywali je Stworzycielami) czyhają na każdy nieostrożny ruch człowieka, aby go połknąć.*

W oddali zamajaczył niewyraźny cień filtranamiotu. Paul podbiegł do niego i w ostatniej chwili wdrapał się na górę. Zęby czerwia uderzyły w piasek...

Czy potrafisz doprowadzić ornitopter młodego władcy do końca piątej strefy otaczającej reaktor atomowy pozostawiony przez barona Harkonnena? Najpierw jednak sięgnij po książkę Franka Herberta „Diuna". Opowiada ona o skomplikowanym systemie rządzenia planetą Diuna, zwaną też Arrakis. O władzę walczą Atrydzi i Harkonnenowie. Książę Leto Atryda w następstwie intryg barona został zamordowany. Jego synowi i lady Jessice pomagają tubylcy — Fremeni. Powieść „Diuna" na pewno zainteresuje każdego miłośnika SF. Frank Herbert napisał dalsze cztery części losów planety, zatytułowane kolejno: „Zbawiciel Diuny", „Dzieci Diuny", „Bóg — władca Diuny", „Heretycy Diuny". Seria ta należy do najwybitniejszych dzieł światowej literatury fantastycznej.

W grze Critical Mass trzeba przelecieć ornitopterem pięć stref ochronnych, dotrzeć do reaktora a następnie rozbić go. Przeznaczone jest na to 10 minut, po upływie których planeta rozpadnie się na kawałki. Zadanie utrudniają sterczące wszędzie skały, szczątki urządzeń do przetworu przyprawy — największego bogactwa Arrakis, a także latające wrogie obiekty. Podczas gry na ekranie znaleźć można informacje o energii, kierunku, w którym znajduje się wróg lub filtrnamiot, odległości od końca strefy i ilości zdobytych punktów. Manewrowanie ornitopterem nie jest łatwe ze względu na jego dużą bezwładność. Spróbuj jednak uratować Arrakis.

(mp)

Firma: Durell Software
Komputer: Commodore 64/128, Spectrum 48/+/128/+2/+3

BAJTKOWI KRÓLOWIE GIER

Radosław Zyskowski, lat 13, uczeń VI klasy Szkoły Podstawowej nr 41 w Łodzi, mieszka w Łodzi przy ul. Rajdowej 9 m 56



Posiadany mikrokomputer: Atari 800XL z magnetofonem
Ulubione gry: Boulder Dash, Spy Vs Spy II
Zainteresowania: Chemia, Informatyka
Wymarzony komputer: Atari 520 ST
Plany na przyszłość: Napisać program edukacyjny z Chemii

Piotr Banaszewski, lat 16, uczeń I klasy L.O. nr 3 w Łodzi, mieszka w Łodzi przy ul. Kusocińskiego 128 m 16.



Posiadany mikrokomputer: Spectrum 48 k
Ulubione gry: Arnhem, Desert Rats
Zainteresowania: Grafika komputerowa
Wymarzony komputer: IBM
Plany na przyszłość: dostać się na Informatykę

POKE rzysta

W nowym Pokerzyście oprócz POKE'ów znalazły się też inne pomoce dla wytrwałych graczy. I tak:

GHOSTBUSTERS — ACTIVISION
TANG BILLY 15570011 — kod na 93600 £

STARSTRIKE II — REALTIME SOFTWARE
HEAK AND OBEY — wpisane po wciśnięciu pauzy a następnie klawisza:
Q — odnowienie paliwa
W — odbudowanie osłony
E — regeneracja lasera

STEINLESS STEEL — MICRO-GEN
ALIK lub SILK — wpisane po wciśnięciu pauzy daje nieśmiertelność i regeneracje osłony

FRANK BRUNO's BOXING — ELITE
kody do załadowania poszczególnych bokserów na imię „STE":
— Fling Long Chop — BS8N8NMAO
— Andra Puchenderow — AMC1NAK9C
— Tribal Trouble — FQ6IN9SN9
— Franchie — IKAIIBQN3
— Ravioli Mafiosi — INDIIAOM6
— Andy Antipodean — NR7IN9MI4
— Peter Perfect — ILBIIOKNI

A teraz zwykłe pomoce w postaci POKE'ów:

LIGHT FORCE — FTL
POKE 40725,0

1942 — ELITE
POKE 47007,255

PAPER BOY — ELITE
POKE 48023,201

WAR — MARTECH
POKE 38394,0

ELITE — FIREBIRD
POKE 46848,201

CYBERUN — ULTIMATE
POKE 36168,175

DAN DARE — VIRGIN
POKE 36268,175 : POKE 45954,104

SPINDIZZY — ELECTRIC DREAMS
POKE 48272,201 : POKE 48401,201

FINDERS KEEPERS — MASTERTRONIC
POKE 34252,0 lub 30394,X : POKE 33969,0 X-il. ludzików

Całe Loadery:

CAULDRON — PALACE SOFTWARE
```
10 REM ...M.1...
20 CLEAR VAL „24599" : FOR n=23296 TO 23309: READ a :
   POKE n,a : NEXT n
30 LET L= USR VAL „23296" : POKE 40060,0 : LET L= USR
   VAL „24600"
40 DATA 221,33,24,96,17,232,159,62,255,55,205,86,201
```

PHOENIX — ALTERNATIVE SOFTWARE
```
10 REM ...M.1...
20 CLEAR VAL „24500": LOAD „„CODE 24532
30 POKE 29375,0 : LOAD „„ CODE
40 RANDOMIZE USR VAL „30105"
```

SWEEVO'S WORLD — GARGOYLE GAMES
```
10 REM ...M.1..
20 CLEAR VAL „24799" : LOAD „„ SCREEN$ . LOAD „„CODE :
   LOAD „„ CODE : POKE 33219,0
30 RANDOMIZE USR VAL „24800"
```

CRITICAL MASS — DURELL SOFTWARE
```
10 REM ...M.1...
20 LOAD „„ SCREEN$ : INK 6 : OVER 1 . PRINT AT 0,0:
30 LOAD „„ CODE : POKE 56879,52
40 PRINT USR VAL „48000"
```

ZUB — MASTERTRONIC
```
10 REM ...M.1...
20 BORDER NOT PI : INK NOT PI : PAPER NOT PI:OVER SGN PI
30 CLEAR VAL „24699"
40 LOAD „„ SCREEN$ : LOAD „„ CODE : LOAD „„ CODE
50 POKE 37473,201
60 RANDOMIZE USR VAL „24700"
```

POKE'i można wprowadzać bezpośrednio do loadera lub na programie COPY-COPY. Loadery i POKE'i mogą nie działać przy innych wersjach programów niż dostępne redakcji.

GRACZ

S.O.S.

Poszukuje dokładnego opisu gry „JEWELS OF BABILON" w wersji na mikrokomputer Amstrad 6128.

Agnieszka Pyka
ul. Polna 44
44-227 Przyszowice

Razem z kolegą gramy już bardzo długo w grę „POP EYE". Niestety, dochodzimy do pewnego momentu z którego nie ma już wyjścia, ponieważ brak jest kluczyka. Pomóżcie.

Presjan Naczew
ul. Wiśniowa 14/28
31-426 Kraków

Poszukuje nieśmiertelności i innych ułatwień w grach: CAULDRON, DYNAMITE DAN, IMP. MISION w wersji na C-64.

Marek Awramienko
ul. Świerczewskiego 30
44-120 Pyskowice

Posiadam grę „TIME-GATE", ale cóż z tego, kiedy jest ona nagrana na kasecie firmowej bez programu ładującego.
Może ktoś z czytelników pomoże mi zdobyć ten program.

Oskar Lewiński
ul. Szkolna 7/7
14-530 Frombork

Poszukujemy informacji na temat gier karate na mikrokomputery Atari 800XL i Spectrum 48 oraz planu gry „SABOTEUR" w wersji na ZX Spectrum.

Marek Suczyk
ul. Helska 5/8
44-100 Gliwice

Jarek Janas
ul. Helska 9/5
44-100 Gliwice

ROZWIĄZYWANIE UKŁADU RÓWNAŃ LINIOWYCH

Rozwiązanie dwóch układu równań z dwoma niewiadomymi, to nie problem. Trzy równania z trzema niewiadomymi, to trochę trudniejsze zadanie. A co zrobić, gdy będzie ich pięć, dziesięć, piętnaście ... W tym przypadku przyda się komputer i ten program. Uwzględnia on także układy, które nie dadzą się jednoznacznie rozwiązać, bądź w ogóle nie posiadają rozwiązania.

Weźmy najprostszy przykład:

$$\begin{aligned} 3x_1+2x_2&=7\\ 5x_1+4x_2&=13 \end{aligned}$$

Ogólnie taki układ (odtąd będę go nazywać układem drugiego stopnia, bo ma dwa równania i dwie niewiadome) można zapisać następująco:

$$\begin{aligned} a_{11}x_1+a_{12}x_2&=b_1\\ a_{21}x_1+a_{22}x_2&=b_2 \end{aligned}$$

gdzie dla naszego przykładu $a_{11}=3$, $a_{12}=2$, $b_1=7$, itd.
Jak wiesz, aby obliczyć x_1 i x_2 wystarczy znać współczynniki „a" i „b".
Załóżmy, że układ jest dowolnego stopnia, np.: n=1, lub n=2, lub innej liczbie naturalnej. Układ taki wygląda następująco:

$$\begin{aligned} a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n&=b_1\\ a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n&=b_2\\ &\vdots\\ a_{n1}x_1+a_{n2}x_2+\ldots+a_{nn}x_n&=b_n \end{aligned}$$

Rozwiążmy układ trzeciego stopnia.

$$\begin{aligned} 1)\quad 2x_1+x_2+3x_3&=13\\ 2)\quad 2x_1+5x_2+9x_3&=39\\ 3)\quad 4x_1+4x_2+12x_3&=48 \end{aligned}$$

Cała sztuka polega na takim mnożeniu, dodawaniu i odejmowaniu stronami, aby uzyskać najprostszą postać równań do obliczenia niewiadomych.
Po pierwsze pozbywamy się współczynników przy x_1 z równań (2) i (3) odejmując od nich stronami równanie pierwsze. Nie ma problemu z równaniem (2), ale równanie (3) ma współczynnik 4 przy x_1, czyli równanie (1) mnożymy przez dwa:

$$\begin{aligned} 1)\quad 2x_1+x_2+3x_3&=13\\ 2)\quad 2x_1+5x_2+9x_3&=39\\ (-)\quad 2x_1+x_2+3x_3&=13 &&\text{równanie 1) mnożę przez 2/2=1 (co oczywiście pomijam)}\\ \hline 4x_2+6x_3&=26\\ 3)\quad 4x_1+4x_2+12x_3&=48 &&\text{równanie 1) mnożę stronami przez 4/2=2}\\ (-)\quad 4x_1+2x_2+6x_3&=26\\ \hline 2x_2+6x_3&=22 \end{aligned}$$

otrzymujemy:

$$\begin{aligned} 1)\quad 2x_1+x_2+3x_3&=13\\ 2)\quad 4x_2+6x_3&=26\\ 3)\quad 2x_2+6x_3&=22 \end{aligned}\quad \Big\}\ \text{podukład przeznaczony do analizy}$$

Teraz pozostawiamy równanie (1) i zajmujemy się tylko nowymi równaniami (2) i (3), czyli analizujemy już układ drugiego stopnia w ten sam sposób jak poprzednio. W tym układzie pozbywam się współczynnika przy x_2 w równaniu (3) poprzez pomnożenie równania (2) przez 2/4=1/2 i odjęcie tak otrzymanej równości od równania (3).

$$\begin{aligned} 1)\quad 2x_1+x_2+3x_3&=13\\ 2)\quad 4x_2+6x_3&=26 &&\text{równanie 2) mnożę stronami przez 2/4=1/2}\\ 3)\quad 2x_2+6x_3&=22\\ (-)\quad 2x_2+3x_3&=13\\ \hline 3x_3&=9 \end{aligned}$$

stąd układ:

$$\begin{aligned} 1)\quad 2x_1+x_2+3x_3&=13\\ 2)\quad 4x_2+6x_3&=26\\ 3)\quad 3x_3&=9 \end{aligned}$$

Widać, że $x_3 = 9/3 = 3$, stąd $x_2 = 26 - 6$

$$\begin{aligned} x_3&=9/3=3\\ x_2&=(26-6x_3)/2=(26-18)/2=2\\ x_1&=(13-(x_2+3x_3))/2=\\ &=(13-11)/2=1 \end{aligned}$$

Dla układu n-tego stopnia wygląda to tak:

$$\begin{aligned} a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n&=b_1\\ a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n&=b_2\\ &\vdots\\ a_{n1}x_1+a_{n2}x_2+\ldots+a_{nn}x_n&=b_n \end{aligned}$$

czyli

$$\begin{aligned} a_{11}x_1+\quad a_{12}x_2+\ldots+\quad a_{1n}x_n&=b_1 &&\text{równanie 1) mnożę stronami przez a21/a11}\\ a_{21}x_1+\quad a_{22}x_2+\ldots+\quad a_{2n}x_n&=b_2\\ (-)\quad a_{21}x_1+(a_{21}/a_{11})a_{12}x_2+\ldots+(a_{21}/a_{11})a_{1n}x_n&=b_1(a_{21}/a_{11})\\ \hline (a_{22}-(a_{21}/a_{11})a_{12})x_2+\ldots+(a_{2n}-(a_{21}/a_{11})a_{1n})x_n&=b_2-b_1(a_{21}/a_{11})\\ a'_{22}x_2\quad+\ldots+\quad a'_{2n}x_n&=b'_2\\ a'_{32}x_2\quad+\ldots+\quad a'_{3n}x_n&=b'_3\\ &\vdots\\ a'_{n2}x_2\quad+\ldots+\quad a'_{nn}x_n&=b'_n \end{aligned}$$

otrzymujemy układ:

$$\begin{aligned} a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n&=b_1\\ a'_{22}x_2+\ldots+a'_{2n}x_n&=b'_2\\ a'_{32}x_2+\ldots+a_{3n}x_n&=b'_3\\ &\vdots\\ a'_{n2}x_2+\ldots+a_{nn}x_n&=b'_n \end{aligned}$$

Omówioną procedurę powtarzam n razy i ostatecznie otrzymujemy układ równań:

$$\begin{aligned} a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n&=b_1\\ a'_{22}x_2+\ldots+a'_{2n}x_n&=b'_2\\ &\vdots\\ a^*_{nn}x_n&=b^*_n \end{aligned}$$

Stąd obliczamy od razu x_n potem x_{n-1}, aż do x_1.
I jeszcze jedna ważna sprawa. Okazuje się, że obliczenia komputerowe są najdokładniejsze gdy a_{11}, a potem a_{22}, itd. są duże co do wartości bezwzględnej, a na pewno różne od 0. Stąd program przeszukuje współczynniki, aby znaleźć największy co do wartości bezwzględnej, a następnie zamienia kolumny i wiersze tak, aby ten współczynnik znalazł się w samym górnym rogu. Jednak zamiana kolumn powoduje zmianę kolejności niewiadomych. Np.: zamiana kolumny z niewiadomymi x_2 i x_3 powoduje, że na miejscu x_2 jest x_3, a na miejscu x_3 znajduje się x_2. Dlatego utworzyłem wektor „x", który wszelkie takie zamiany pamięta. Ale jak to działa spróbuj rozszyfrować sam.

Ponieważ przyjąłem, że układ ma stopień równy n, to musiałem utworzyć tablice: n(n,n+1), y(n), w(n), x(n). W tablicy n(n,n+1) umieściłem współczynniki w następującej kolejności:

a_{11}	a_{12}	...	a_{1n}	b_1
a_{21}	a_{22}	...	a_{2n}	b_2
...	...	...	...	...
a_{n1}	a_{n2}	...	a_{nn}	b_n

Tablica x zapamiętuje kolejność przestawień kolumn, aby odtworzyć potem prawidłowe ustawienie zmiennych $x_1, \ldots, x_n$. Tablica w zawiera ostatecznie obliczone wartości zmiennych $x_1, \ldots, x_n$, a tablica y służy do obliczenia niewiadomych.
Jeśli, czego Ci nie życzę, nastąpi STOP w linii 1100 to oznacza, że układ nie jest rozwiązywalny.
Niestety musisz wprowadzić współczynniki do tablicy n. Dlatego proponuję Ci napisać następujący program główny:
(Wydruk I).
Dla przykładu wprowadź układ:

$$\begin{aligned} 2x_1+3x_2+4x_3&=20\\ x_1+2x_2+8x_3&=29\\ 5x_1+3x_2+4x_3&=23 \end{aligned}$$

czyli $a_{11}=2$, $a_{11}=3$, itd.
Otrzymane wyniki powinny wynosić: $x_1=1$
$x_2=2$
$x_3=3$
Dla przykładu proponuję przeanalizować układ:

$$\begin{aligned} 2x_1+3x_2+4x_3&=20\\ x_1+2x_2+8x_3&=29 \qquad ?!\\ 4x_1+6x_2+8x_3&=40 \end{aligned}$$

Oczywiście jest on nierozwiązywalny jednoznacznie.
A teraz coś poważnego. Z równym powodzeniem można uporać się z układem równań w dziedzinie liczb zespolonych. Np. układ drugiego stopnia wygląda następująco:

$$\begin{aligned} (r_{11}+jz_{11})(w_1+jt_1)+(r_{12}+jz_{12})(w_2+jt_2)&=v_1+jy_1\\ (r_{21}+jz_{21})(w_1+jt_1)+(r_{22}+jz_{22})(w_2+jt_2)&=v_2+jy_2 \end{aligned}$$

Jest to ukryty zapis układu czwartego stopnia:

$$\begin{aligned} r_{11}w_1+r_{12}w_2-z_{11}t_1-z_{12}t_2&=v_1\\ r_{21}w_1+r_{22}w_2-z_{21}t_1-z_{22}t_2&=v_2\\ z_{11}w_1+z_{12}w_2+r_{11}t_1+r_{12}t_2&=y_1\\ z_{21}w_1+z_{22}w_2+r_{21}t_1+r_{22}t_2&=y_2 \end{aligned}$$

Spróbuj sam odpowiedzieć na dwa pytania: Dlaczego tak można zapisać te równania i w jaki sposób działa drugi program główny (Wydruk II)? Zwróć uwagę na to, że jest to trochę zmodyfikowany program pierwszy.
Dla sprawdzenia rozwiąż równanie drugiego stopnia:

$$\begin{aligned} (1+j3)(w_1+jt_1)+(2+j4)(w_2+jt_2)&=-20+j22\\ (2+j\ )(w_1+jt_1)+(3+j2)(w_2+jt_2)&=-3+j23 \end{aligned}$$

Wyniki końcowe: w(1)=1
t(1)=3
w(2)=2
t(2)=4

Krzysztof Poźniak
(Klub HOBBYTE)

Listing 1

```
10 INPUT "STOPIEN UKLADU n=";n
20 DIM n(n,n+1)
30 DIM x(n)
40 DIM y(n)
50 DIM w(n)
60 FOR y=1 TO n
70 FOR x=1 TO n
80 PRINT "a(",y;",",x,")=" INPUT n(y,x)
90 NEXT x
100 PRINT "b(";y;")=":INPUT n(y,n+1)
110 NEXT y
120 FOR k=1 TO n
130 LET x(k)=k
140 NEXT k
150 GOSUB 1000
170 FOR k=1 TO n
180 PRINT "x(";k;")=";w(k)
190 NEXT k
200 STOP
```

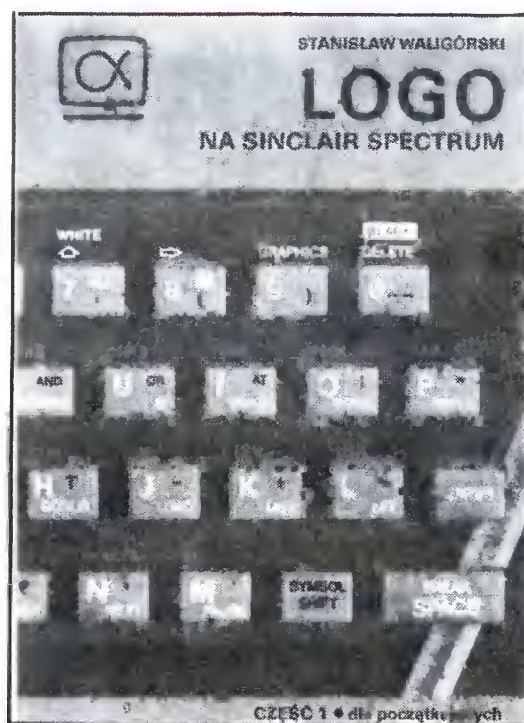
Listing 2

```
10 INPUT "STOPIEN UKLADU n=";n
20 DIM n(2*n,2*n+1)
30 DIM x(2*n)
40 DIM y(2*n)
50 DIM w(2+n)
60 FOR y=1 TO n
70 FOR x=1 TO n
80 PRINT "r(";y,",";x;")=":INPUT n(y,x)
82 LET n(n+y,n+x)=n(y,x)
85 PRINT "z(";y;",";x;")=":INPUT n(n+y,x)
87 LET n(y,n+x)=-n(n+y,x)
90 NEXT x
100 PRINT "v(";y;")=":INPUT n(y,2*n+1)
105 PRINT "y(";y;")=":INPUT n(n+y,2*n+1)
110 NEXT y
120 FOR k=1 TO 2*n
130 LET x(k)=k
140 NEXT k
145 LET n=2*n
150 GOSUB 1000
170 FOR k=1 TO n/2
180 PRINT "w(";k;")=";w(k)
185 PRINT "t(";k;")=";w(n/2+k)
190 NEXT k
200 STOP
```

Listing 3

```
1000 FOR k=1 TO n
1010 LET LAX=0
1020 FOR y=k TO n
1030 FOR x=k TO n
1040 IF ABS (n(y,x))<=ABS (LAX) THEN GOTO 1080
1050 LET LAX=n(y,x)
1060 LET MY=y
1070 LET MX=x
1080 NEXT x
1090 NEXT y
1095 REM jesli LAX=0 uklad nieoznaczony
1100 IF LAX=0 THEN STOP
1110 FOR l=1 TO n+1
1120 LET ZMPOM=n(k,l)
1130 LET n(k,l)=n(MY,l)
1140 LET n(MY,l)=ZMPOM
1150 NEXT l
1160 FOR l=1 TO n
1170 LET ZMPOM=n(l,k)
1180 LET n(l,k)=n(l,MX)
1190 LET n(l,MX)=ZMPOM
1200 NEXT l
1210 LET ZMPOM=x(k)
1220 LET x(k)=x(MX)
1230 LET x(MX)=ZMPOM
1240 REM element maksymalny jest ustawiony
 w lewym gornym rogu podukladu-uzyskuj
emy wtedy najwieksza dokladnosc obliczen
1250 FOR l=k+1 TO n
1260 LET MNOZ=n(l,k)/n(k,k)
1270 FOR j=k+1 TO n+1
1280 LET n(l,j)=n(l,j)-n(k,j)*MNOZ
1290 NEXT j
1300 NEXT l
1310 NEXT k
1320 FOR k=n TO 1 STEP -1
1330 LET suma=0
1340 FOR l=k+1 TO n
1350 LET SUMA=SUMA+y(l)*n(k,l)
1360 NEXT l
1370 LET y(k)=(n(k,n+1)-SUMA)/n(k,k)
1380 NEXT k
1390 FOR k=1 TO n
1400 LET w(x(k))=y(k)
1410 NEXT k
1420 REM tablica W zawiera wyniki
1430 RETURN
```

WARTO PRZECZYTAĆ



Nareszcie użytkownicy mikrokomputerów, chcący nauczyć się posługiwania językiem LOGO, przestają być skazani na mozolne wyszukiwanie cennych informacji rozproszonych w wielu numerach komputerowych czasopism. „Logo na Sinclair Spectrum dla początkujących" Stanisława Waligórskiego jest bowiem pierwszą — lecz już wiadomo, że nie ostatnią — pozycją książkową traktującą o tym popularnym języku.

Autora nie trzeba czytelnikom „Bajtka" przedstawiać — był on przewodniczącym zespołu opracowującego program „elementów informatyki" w szkołach i jest współtwórcą polskiej wersji LOGO (patrz wywiad w n-rze 9/86 „Bajtka"). Stąd treść liczącej niespełna 60 stron książeczki napisana jest w pewnej mierze pod kątem użycia LOGO w szkole; nie znaczy to jednak, że prywatny użytkownik komputera nie znajdzie tu niczego dla siebie.

Książka przeznaczona jest również dla tych, którzy stykają się z komputerem po raz pierwszy: część tekstu poświęcono zasadom obsługi Spectrum i sposobowi wczytywania LOGO z magnetofonu. W dalszej kolejności czytelnik jest zapoznawany z kolorową grafiką żółwia i metodami posługiwania się nią. Tę część książki — z wyjątkiem może rozdziału poświęconego kolorom — można polecić również użytkownikom Atari lub Amstradów.

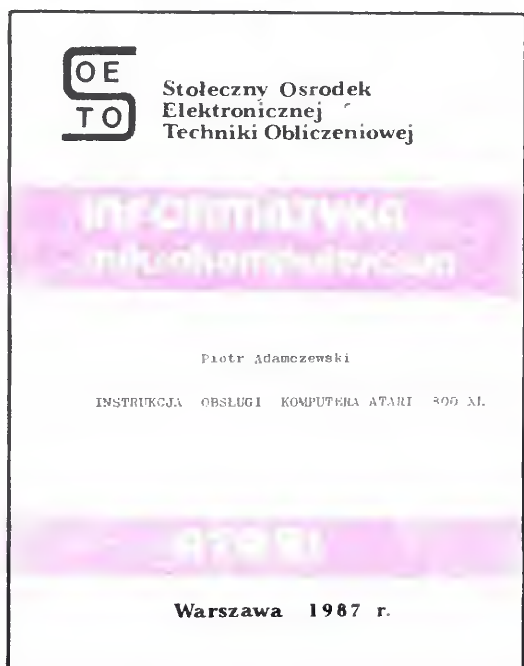
Nie jest to jednak — co warto podkreślić — tylko kurs języka programowania. Wiele uwagi poświęcono w książce metodom rozwiązywania problemów, i to już na etapie pierwszych prostych ćwiczeń.

Całość dotyczy angielskiej wersji LOGO, lecz na końcu książeczki zamieszczono słowniczek polskiej i angielskiej wersji języka. Dodatkowo mieści się tam opis wszystkich komend wraz z objaśnieniami.

Wadą publikacji jest przede wszystkim niewielka objętość. Wiedzę zawartą w książce można posiąść w trzy dni, a co dalej? Na szczęście w niedługim czasie zapowiedziana jest część druga — „Logo na Sinclair Spectrum dla zaawansowanych".

(mb)

Stanisław Waligórski „Logo na Sinclair Spectrum. Część 1 — dla początkujących" Wyd. I, IWZZ Warszawa 1987; s. 56, nakład 50 tys. egz.



Instrukcja **obsługi mikrokomputera Atari 800 XL**

Dotychczas jedynym polskim podręcznikiem dla użytkowników komputerów Atari był skrypt „Atari BASIC" pod redakcją Wiesława Miguta. Zapowiadane od dłuższego czasu przez KAW wydanie książkowe jeszcze nie doszło do skutku. Wykorzystało tę lukę warszawskie SOETO wydając (także jako skrypt) „Instrukcję obsługi mikrokomputera Atari 800 XL", która powinna się znaleźć w bibliotece każdego posiadacza tego komputera.

Książka składa się z czterech części. Część pierwsza zawiera opis komputera, sposób jego podłączenia i eksploatacji oraz działanie klawiatury ze szczególnym uwzględnieniem klawiszy funkcyjnych i specjalnych. Na końcu tej części znajduje się opis użycia komputera jako prostego kalkulatora. Część druga jest dokładnym opisem instrukcji wbudowanego interpretera języka BASIC oraz ich składni, popartym licznymi przykładami. Część trzecia stanowi wprowadzenie do programowania w języku maszynowym mikroprocesora 6502. Zawiera listę rozkazów procesora oraz ich krótkie wyjaśnienie. Czwarta część jest złożona z trzech dodatków uzupełniających treść książki oraz opisu podstawowych urządzeń peryferyjnych komputera Atari.

Książka jest napisana przystępnym językiem, który oszczędnie operuje zwrotami technicznymi i jest dzięki temu łatwo zrozumiała nawet dla początkującego czytelnika. Wiadomości i informacje techniczne o budowie wewnętrznej są podane w ilości niezbędnej do zrozumienia zasad obsługi komputera. Za zbyt pobieżną można uznać jedynie część dotyczącą języka maszynowego, lecz jest to usprawiedliwione przeznaczeniem podręcznika dla początkujących użytkowników.

Należy stwierdzić, że wydanie tej książki zapełni dotkliwą lukę na naszym rynku wydawniczym, a jej poziom merytoryczny spowoduje ograniczenie napływu licznych tłumaczeń sprzedawanych przez różne firmy prywatne po astronomicznych cenach. Inicjatywa ta jest tym cenniejsza, że komputery firmy Atari rozprowadzane przez „Pewex" zdobyły w Polsce ogromną popularność, brak jest natomiast dotyczącej ich literatury.

Książka jest do nabycia w siedzibie SOETO — Warszawa, ul. Hoża 50.

(ziew)

Piotr Adamczewski — „Instrukcja obsługi mikrokomputera Atari 800 XL", Warszawa 1987, Stołeczny Ośrodek Elektronicznej Techniki Obliczeniowej. Wydanie I. Nakład 5 tys. egz. Cena ok. 600 zł.

KRZEMOWE I SZARE

Komputery zajmują nas dziś w równym stopniu jak kosmonautyka w latach 60-tych.

Kosmonautyka dotyczyła jednostek, a pisali i mówili o niej wszyscy. Kosmonautyka ogniskowała wyobraźnię pisarzy należących do klanu science fiction. Ale dopiero kiedy poleciał w kosmos Hermaszewski zaczęliśmy mówić na serio o przeciążeniach, o odwapnieniu kości... Okazało się więc, że mieliśmy o kosmonautyce częściowo fałszywe wyobrażenia ponieważ dotyczyła nas zaledwie pośrednio.

Nie mówię, że dokładnie tak jest dziś z komputerami. W przeciwieństwie do rakiet kosmicznych komputery osobiste są w powszechnym użyciu. Ale jeszcze powszechniejszym jest teflon, który zawdzięczamy kosmonautyce. Natomiast mimo setek tysięcy komputerów w Polsce nie korzystamy z sieci komputerowej, nie mamy skomputeryzowanych banków (nie wewnętrznie lecz między sobą), nie mamy komputerowych bez danych. Komputer może stać w moim mieszkaniu, mogę łupać na nim pięć godzin dziennie w wyrafinowaną grę „Chodzi lis koło nory", a mimo to być równie odległym od komputeryzacji, jak od rakiet stojących w Bajkonurze i na Przylądku Kennedy'ego.

Znam parę dobrych opowiadań fantastycznych o komputerach. Rzecz znamienna, wszystkie one mają tonację przestrzegającą. Ale jednocześnie właściwie żadne z opowiadań nie traktuje o immanentnym fataliźmie zawierającym się w maszynie, lecz o kraksie rodzącej się na styku komputera i człowieka. O kraksie wynikającej z niedopasowania bądź człowieczej intrygi. „NACIŚNIJ ENTER" Varley'a mówi o policyjno-wojskowym osaczeniu jednostki przez sieć komputerową. Komputer jest tu narzędziem inwigilacji i zniewolenia. W „Wynajętym człowieku" Baranieckiego to ludzie wprowadzili do komputera pokładowego program szaleńczej gry o wszystko, w wyniku którego to ludzie będą się musieli zabijać. Jest takie stare opowiadanie Zajdla „Dyżur" — do naukowca zatrudnionego w stacji komputerowej dzwonią interesanci i w momencie awarii systemu ten dyżurny naukowiec rozwiązuje zadanie za komputer na zwykłym suwaku. Wprawia to zleceniodawcę w osłupienie. Jest wreszcie wspaniałe opowiadanie o komputerze pracującym w składnicy księgarskiej i o sporze między nim a jednym z nabywców, co doprowadza do skazania owego nabywcy na karę śmierci za kidnaping. Jedynym grzechem owego nabywcy była niechęć do płacenia za zdefektowany egzemplarz... „Porwanego za młodu" Stevensona... Człowiek osaczony, oddający się niebezpiecznej zabawie, człowiek dobrowolnie ogłupiony — oto co, według pisarzy fantastów, może nam zagrozić w wyniku upowszechnienia się komputerów. Wiem, że to bajki, ale w każdej bajce jest ziarnko mądrości.

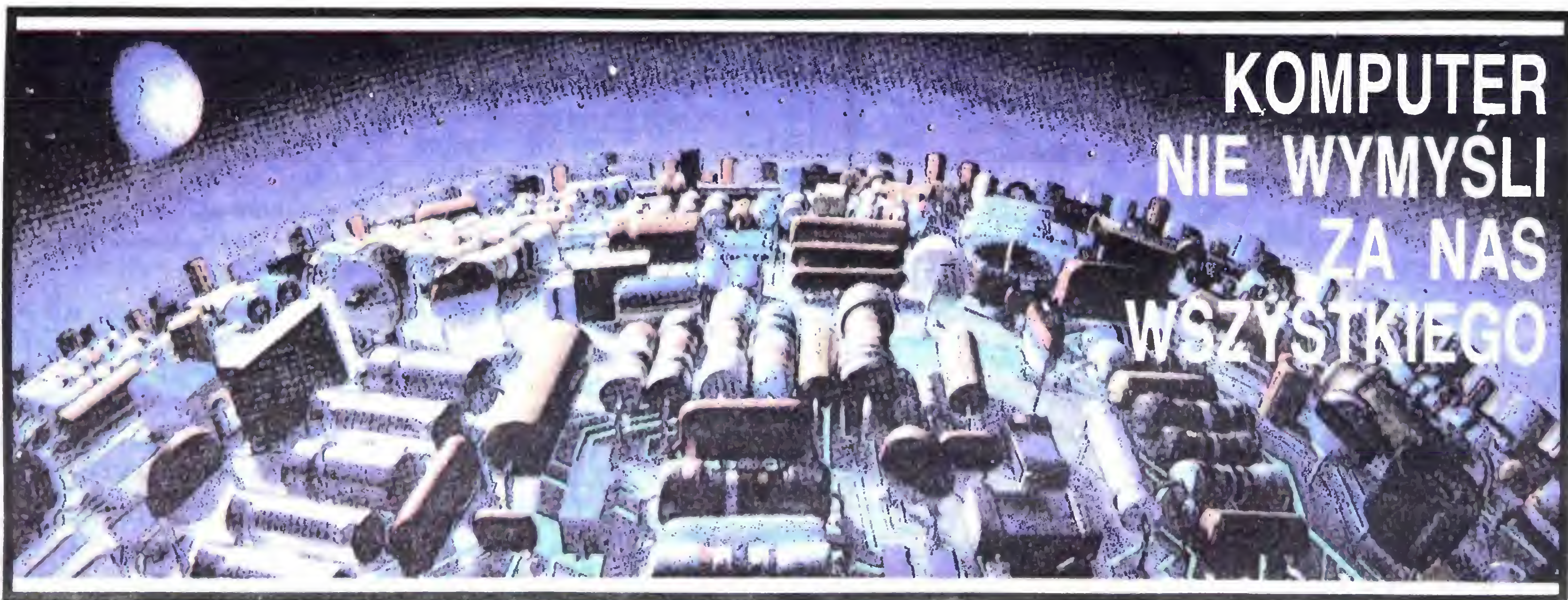
Nie za często i niezbyt pilnie przeglądam „Bajtka", a potem rozmawiam ze szczęśliwymi posiadaczami komputerów. Jak Polska długa i szeroka rozlega się wołanie o program. Trochę mnie to przeraża. I w „Bajtku" i w „Komputerze" przynajmniej co drugi numer, może delikatnie ale zawsze, może zbyt drobną czcionką, ale na tyle dużą, że można ją odczytać, sączycie niezbędne zastrzeżenie — komputer nie zrobi, nie obejmie, nie wymyśli za nas wszystkiego. Jest tym dla mózgu czym rower dla ciała (to bodaj z W. Siwińskiego) — pomoże się poruszać, ale trzeba pedałować.

Trzeba pedałować — zastrzeżenie pierwsze. Nie można na rowerze załatwić wszystkiego — zastrzeżenie drugie. Nie można oczekiwać, że rower pofrunie — oto bardzo ważne zastrzeżenie trzecie. Wszystkie te analogie są oczywiście kulawe, bo ktoś tam przeleciał przez La Manche właśnie na latawcu napędzanym pedałami. No dobrze, ale rowery w dalszym ciągu nie służą do latania. Do czego natomiast nie służą komputery?

Powiedziałem — rozlega się wołanie o program. Słyszę w nim skowyt niebezpiecznej, bo nigdy nie zaspokojonej pożądliwości — wymyślcie, skonstruujcie, zestawicie mi program na wszystko. Tu już nie chodzi o gry, o ładne grafiki, o podręczny katalog książek, domową encyklopedię, o budżet, terminarzyk domowy... Tu chodzi o program na życie. Nie ma i nie będzie takiego programu.

Naprawdę, sytuacja jest dziwna. Człowiek — istota myśląca, użytkująca zaledwie kilkanaście procent mocy swego mózgu, buduje urządzenie, które go z tej pracy jeszcze odciąży. To znaczy inny człowiek buduje, a inny się odciąża. Człowiek istota wielowymiarowa — myśląca, fizyczna i metafizyczna, estetyczna, uczuciowa, obdarzona zdolnością odróżniania dobra od zła — redukuje swój wielowymiarowy język intelektualny i duchowy, swoją różnorodną technikę porozumiewania się przy pomocy słów, ale także gestów, intonacji, grymasów... do kodu elektronicznej maszyny. Wręcz dostosowuje swój bogaty język do prostego języka maszyny i czyni z tego cnotę. Istota, która lat temu dziesięć, dwadzieścia, trzydzieści przeżywałaby różnorodne fascynacje lekturowe, filmowe, muzealne, teatralne, uczuciowe, ale i filozoficzne — dziś doznaje pełnej satysfakcji i intelektualnej szczęśliwości kreśląc na nieostrym ekranie drgające bryły, bądź łupiąc w klawiaturę przy okazji gry w wyjmowanie pierścienia z zaczarowanej komnaty i w zestrzeliwanie przybyszów z kosmosu.

Trochę nie wiem jak z nimi rozmawiać. Kiedy próbuję im mówić o ograniczeniach czy zastrzeżeniach Churcha, Godla i Turinga, zarzucają mi wstecznictwo i oświadczają, że facet taki jak ja unieszczęśliwi syna. Odpowiadam bardzo poważnie, że syn aktualnie uczeń klasy VIII-ej, przerabia „Pana Tadeusza" i że do zrozumienia tragicznych tonów Epilogu, do przeniknięcia cynicznej manipulacyjno-politycznej gry jaką uprawia Gerwazy w scenach Rady, komputer mojemu synowi nie przyda się na nic. Fanatycy komputerowi nie podejmują tematu, ale natychmiast oświadczają, że zmieniłbym poglądy, gdybym sobie sprawił WORDPROCCESSOR. Odpowiadam krótką kalkulacją kosztów (ekran + drukarka + komputer) i podliczam skromną liczbę stron jakie zaczerniam w ciągu miesiąca. Dodaję jeszcze, że lubię szelest długopisu po maszynopisie i te rodzące się w głowie pomysły i poprawki kiedy po raz kolejny przepisuję pokreślony tekst, a nie wtedy, gdy go za mnie wystukuje drukarka. Mówię wtedy, że jestem prymitywem. Odpowiadam by sprawili wordproccessor Szekspirowi i Mickiewiczowi. Po czym, ponieważ atmosfera robi się gorąca próbujemy zadzwonić (od sąsiadów lub z połykającego monety automatu) do którejś z najbliższych knajp, żeby tam się pogodzić przy stoliku pod palmą, bądź do kina, żeby zarezerwować miejsce w 7-mym rzędzie na „Elektronicznego mordercę". Niestety — telefon nie odpowiada. Tak oto się bawimy i to jest moja przewaga, bo przecież bawimy się bez komputera. Kiedy to podkreślam



wpadają we wściekłość, a jeden z nich rzuca we mnie nawet przepalonym joystickiem.

I wtedy, uderzony w głowę, doznaję porażenia odkryciem, że oto ja, człowiek przednówka XXI wieku, nad którego głową fruwają setki satelitów komunikacyjnych — nie mogę dodzwonić się do kina odległego o parę kilometrów. Wracam więc do porównania między erą kosmiczną przed trzydziestu laty a komputerową dzisiaj: Kosmonautyka dotyczyła jednostek a pasjonowali się nią wszyscy, komputeryzacja mogłaby dotyczyć wszystkich, ale dotyczy narazie tylko jednostek (z personalnymi komputerami). Miliardy kilobajtów są zatrudniane w Polsce każdego dnia, bez żadnego ogólnego pożytku.

Pyta mnie redakcja „Bajtka", jak sobie wyobrażam komputeryzację u nas w roku 2000? Nie bardzo ją sobie wyobrażam, bo jestem facetem pozbawionym wyobraźni. Umiem natomiast marzyć, jak wszyscy. Marzenia prowadzą mnie ku wizji manii komputerowej nieco spokojniejszej i nieco lepiej zorganizowanej i spożytkowanej. Marzą mi się komputery stojące za nas w kolejkach (mówię to metaforycznie rzecz jasna), strzegące punktualności pociągów i samolotów, nie mylące się przy okazji przy wypłacaniu rent i wystawianiu rachunków za gaz; marzą mi się komputerowe bazy danych sprzężonych z ogólnodostępnymi kserokopiarkami, komputerowe konta (z wymienialną złotówką), komputerowe podręczne turystyczne translatory foniczno-tekstowe, komputerowe biura podróży, komputerowy gaźnik w... maluchu! Szczerze mówiąc trudno mi nawet marzyć, bo z trudnością nadążam za tym co bywa wdrażane w Japonii i na Zachodzie.

Moje duchowe skłonności prowadzą mnie ku innym jeszcze kwestiom, filozoficznym. Otóż marzy mi się, że ogólna komputeryzacja, jaka bez wątpienia nastąpi, nie pociągnie za sobą wymogu redukcji człowieczeństwa do poziomu jaki najbardziej odpowiada maszynie. Dlatego czekam na moment, w którym powszechnie będzie już uświadomiona prawda, że najlepszy program nie wyjaśnia wszystkiego, że są rzeczy z dziedziny ducha (poezji, etyki, estetyki, religii, dobra), które w nikłym tylko stopniu będziemy mogli zaczerpnąć z mózgów elektronowych, jeśli im w odpowiednim momencie nie przygotujemy miejsca we własnych głowach i sercach.

Tak więc marzy mi się, że cała gramatyka myślenia maszynowego jaką teraz przerabiamy, nie tylko wymości nam komputerową wygodą przyszłe życie, lecz zbliży nas do zrozumienia fenomenu jakim jest myślenie i przeżywanie człowieka. Kiedy to nastąpi będziemy mogli wdrożyć jedyny właściwy program jaki można sobie wyobrazić. Będzie to program wdrażania komputeryzacji nie zaborczej, znającej swoje granice, a więc komputeryzacji kompatybilnej do CZŁOWIEKA.

Maciej Parowski

EFEKTYWNOŚĆ I ELEGANCJA (1)

Kontynuujemy przegląd podstawowych elementów techniki pisania programów, mających wpływ na efektywność otrzymanego kodu.

Zaczniemy od przykładu, w którym wyjątkowo uda się zaoszczędzić i czasu i pamięci. Poniższy program nadaje wartości początkowe elementom dwóch tablic T i A:

```
PROGRAM P6
FOR I=1 TO N
  T(I)=0
NEXT I
FOR I=1 TO N
  A(I)=I
NEXT I
```

Wykonanie pętli to nie tylko wykonanie N razy zawartych w niej instrukcji (w naszym przypadku podstawienia) ale również wykonanie tyle samo razy operacji pomocniczych, służących do zorganizowania pętli, takich jak zwiększenie wartości zmiennej I o 1, sprawdzenie czy już I równa się N itd. Jednym słowem organizacja pętli też kosztuje — widać to wyraźnie jeśli organizujemy tę pętlę sami, przy pomocy skoków warunkowych. W takim razie, skoro już jedną pętlę zorganizowaliśmy, to wykorzystajmy ją do maksimum.

```
FOR I=1 TO N
  T(I)=0
  A(I)=I
NEXT I
```

W ten sposób skracamy również program o kod potrzebny na zorganizowanie drugiej pętli.

Skoro już mowa o minimalizacji kosztów organizacji pętli, to popatrzmy na następujące instrukcje:

```
FOR I=1 TO N*M
  T(I)=I
NEXT I
```

Żeby sprawdzić, czy I osiągnęło już końcową wartość trzeba po każdym wykonaniu zawartości obliczyć wartość wyrażenia N*M. I jest to obliczenie niepotrzebne, bo wartości N i M nie zmieniają się wewnątrz pętli, więc można zrobić tak:

```
NM = N*M
FOR I=1 TO NM
```

itd.

Przejdźmy teraz do następnego zagadnienia, którym jest wykorzystanie w programach tzw. zmiennych indeksowanych, czyli odwołań do elementów tablic. Są one kosztowniejsze niż odwołania do zwykłych zmiennych, gdyż wymagają obliczenia położenia w pamięci żądanego elementu tablicy na podstawie podanych indeksów. Rozważmy następujące zadanie: mamy dane współczynniki A, B, C trójmianu kwadratowego, czyli funkcji $A*x^2 + B*x + C$. W tablicy X są wartości x, dla których trzeba obliczyć wartość trójmianu. Można to zrobić tak:

```
PROGRAM P7
FOR I=1 TO N
  PRINT A*X(I)*X(I) + B*X(I) + C
NEXT I
```

trzy razy odwołując się do tego samego elementu X(I). Można również użyć zmiennej pomocniczej, która pozwoli tego uniknąć:

```
PROGRAM P8
FOR I=1 TO N
  POM=X(I) REM czy to podstawienie musi być w pętli?
  PRINT A*POM*POM + B*POM + C
NEXT I
```

W naszym przykładzie użyliśmy tablicy zawierającej liczby, jednak takie podejście stosuje się do tablic wszystkich typów.

Następnym godnym uwagi źródłem oszczędności są wyrażenia arytmetyczne. Obliczmy dwa pierwiastki trójmianu kwadratowego

```
X1 = (-B - SQRT (B*B - 4*A*C)) / (2*A)
X2 = (-B + SQRT (B*B - 4*A*C)) / (2*A)
```

Dwa razy obliczana jest dokładnie ta sama wartość pierwiastka. Można to łatwo usprawnić, wykonując obliczenie pomocnicze i zapamiętując jego wynik:

```
PROGRAM P9
SQ = SQRT (B*B - 4*A*C)
X1 = (-B - SQ) / (2*A)
X2 = (-B + SQ) / (2*A)
```

Jest to usprawnienie tym bardziej ważne, że obliczanie wartości funkcji matematycznych, takich jak pierwiastek, logarytm, sinus itd. wymaga wykonania przez komputer dużej liczby obliczeń, a więc zajmuje wielokrotnie więcej czasu niż zwykłe działania arytmetyczne czy podstawienie wartości na zmienną. Jednak zapamiętywanie na zmiennych pomocniczych tych fragmentów wyrażeń arytmetycznych, które powtarzają się w programie wielokrotnie, jest opłacalne nawet gdy nie występują w nich wywołania funkcji.

W programie P9 możemy jeszcze „pozbyć się" jednej operacji dzielenia:

```
POM = 1/(2*A)
SQ = SQRT (B*B - 4*A*C)
X1 = (-B - SQ) *POM
X2 = (-B + SQ) *POM
```

Zrobiliśmy to kosztem użycia dodatkowej zmiennej i... zmniejszenia się czytelności programu.

Ważna jest umiejętność przekształcania wyrażeń arytmetycznych. W programie P8 możemy usunąć z pętli jeszcze jedno mnożenie tylko dzięki wyciągnięciu przed nawias — popatrzcie uważnie.

Oszczędność czasu wykonania można uzyskać również dzięki właściwej budowie programu. Znowu posłużymy się przykładem:

```
PROGRAM P10
IF DTYG=1 THEN PRINT "poniedziałek"
IF DTYG=2 THEN PRINT "wtorek"
IF DTYG=3 THEN PRINT "środa"
```

Z trzech powyższych warunków tylko jeden może być prawdziwy, niemniej jednak, nawet jeśli prawdziwy jest już pierwszy, to i tak sprawdzane będą wszystkie! Możemy tego uniknąć jeśli język, w którym programujemy rozpoznaje konstrukcję **IF... THEN ... ELSE.** Możemy wtedy napisać:

```
PROGRAM P11
IF DTYG=1 THEN
  PRINT "poniedziałek"
ELSE
  IF DTYG=2 THEN
    PRINT "wtorek"
  ELSE
    IF DTYG=3 THEN
      PRINT "środa"
    ENDIF
  ENDIF
ENDIF
```

Niektóre języki programowania, np. PASCAL, zawierają specjalną instrukcję CASE, która pozwala zapisać to samo co zawiera program P11 w dużo wygodniejszy sposób.

Ważnym elementem programów są procedury, lub jak kto woli podprogramy. Instrukcja wywołania podprogramu jest instrukcją dość kosztowną, przy czym koszt jest tym większy im więcej parametrów przekazujemy do wołanego podprogramu. Jednak upatrywanie źródła oszczędności w eliminacji podprogramów było by dużym błędem. Rozsądnie dobrane podprogramy są elementem praktycznie niezbędnym przy realizacji prawie wszystkich zadań, a zyski płynące z ich używania zawsze przekraczają koszty wywołań. Pamiętajmy tylko, że jeśli nie jest to konieczne, to nie należy umieszczać wywołań wewnątrz pętli.

Na zakończenie chciałbym jeszcze raz przypomnieć, że jak w każdej dziedzinie życia, tak i w programowaniu potrzebny jest zdrowy rozsądek. Chciałbym, aby podane wyżej przykłady i zasady posłużyły Wam do zrozumienia pewnych mechanizmów występujących przy realizacji programów przez współczesne komputery. Natomiast potraktowanie podanego materiału jako zbiór zasad, których należy ślepo i bezmyślnie przestrzegać może w szczególnych przypadkach przynieść więcej strat niż pożytku.

Pamiętajmy również, że programu zbudowanego w oparciu o nieefektywny, za wolny algorytm nie uratują żadne sztuczki programistyczne, oraz o tym, że nadrzędną wartością dla programów jest poprawne działanie. Program w pełni zoptymalizowany, ale działający niepoprawnie jest zupełnie bezwartościowy.

Andrzej Pilaszek

WSZYSTKO DLA WSZYSTKICH

Informujemy, że nastąpiła zmiana cen za ogłoszenia zamieszczane w Bajtku.

Aktualnie cena reklamy biało-czarnej wynosi 300 zł za 1 cm². Do ceny podstawowej doliczane jest 30% za dodatkowy kolor i 100% w przypadku reklamy wielobarwnej.

Ogłoszenie drobne kosztuje 200 zł za jedno słowo.

Redakcja nie ponosi odpowiedzialności za treść ogłoszeń

GIEŁDA (ceny na dzień 87.05.23)

Giełdy Bajtka odbywają się w Warszawie przy ul. Grzybowskiej 35 (Szkoła Podstawowa nr 25) w każdą sobotę od 14 do 19 oraz w każdą niedzielę od 10–16 przy ul. Saskiej 78 (Technikum Chemiczne).

	GIEŁDA BAJTKA (tys. zł)	PEWEX BALTONA (USD)	RFN (śred) (DM)
SINCLAIR			
ZX 81	40–50	—	39
ZX Spectrum 48 KB	110	115	150–200
ZX Spectrum Plus	125–140	—	180–280
ZX Spectrum 128 + 2	240	—	—
Drukarka SEIKOSHA GP 50S	80–120	—	199
TIMEX 2048	120	146	—
Joystick QUICKSHOT II	9–11	—	9–15
COMMODORE			
C-64 (nowa wersja)	225	219	370–449
C-128	390–430	299	590
C-128D	850	—	1199
Amiga z monitorem kolorowym	—	—	2900
Magnetofon 1531	35–40	48	49–65
Stacja dyskietek 1541	230–250	—	450
Stacja dyskietek 1571	310	299	490–540
Drukarka MPS 801	200–240	—	199
Dyskietki 5 1/4 (średnia jakość)	0.65–1.5	3.5	0.5–1.8
ATARI			
65XE	140	125	40–180
130 XE	205–220	199	360
Stacja dyskietek 1050	210	187	370
Drukarka 1029	240	199	299
ATARI 520 STM st. dysk. 0.5Mb	800–900	—	970
AMSTRAD			
464 z monit. monochromat.	290	—	520
6128 z monit. monochromat.	510	—	900
6128 z monitorem kolorowym	600	—	1250
PCW 8256	—	—	1500
Dyskietki 3	5–6.5	—	7–12
Stacja dyskietek 3 do 464	195	—	520
PC 1512 SD	—	—	1189

HISTORIA ROZWOJU ARCHITEKTURY KOMPUTERÓW

Niewątpliwie, najważniejszym motorem postępu jest lenistwo. Ono bowiem powoduje, że wymyślamy wciąż nowe rzeczy po to, by wykonanie jakiejś pracy wymagało jak najmniej wysiłku. Nic więc dziwnego, że z chwilą powstania komputerów, człowiek starał się obarczyć te potulne maszyny możliwie największą ilością pracy.

Najpierw człowiekowi nie spodobało się, że procesor komputera (odpowiednik naszego mózgu) stoi bezczynnie, oczekując na wykonanie operacji wprowadzenia lub wyprowadzenia danych. Skonstruowano więc kanały transmisyjne, które przejęły funkcje sterowania przepływem informacji. Procesor miał tylko zainicjować pracę kanału i mógł przystąpić do realizacji innego zadania.

Po zakończeniu transmisji, kanał wysyłał do procesora sygnał, po otrzymaniu którego, procesor przerywał swoje aktualne zadanie (stąd nazwa sygnału — przerwanie) i reaktywował zadanie, w którym zainicjowano zakończoną przed chwilą transmisję danych. Dzięki takiemu systemowi przerwań komputer mógł realizować kilka zadań, pozornie jednocześnie. Następny program mógł zacząć się wykonywać zanim poprzedni zakończył swoje działanie. W konsekwencji, każdy program czekał znacznie krócej w kolejce do procesora.

Wszystkie, aktualnie wykonywane programy musiały być jednocześnie w pamięci operacyjnej. Zaspokojenie potrzeb komputera i wyposażenie go w odpowiednio dużą pamięć operacyjną spowodowałoby niewspółmierne do efektów, zwiększenie kosztów budowy komputerów.

W tym momencie ujawnił się kolejny motor postępu — skąpstwo. Człowiek chciałby osiągnąć cel płacąc za to możliwie najniższą cenę. Prawdopodobnie dlatego pierwszy człowiek zaczął chodzić na dwóch nogach, żeby jego potomkowie wydawali mniej pieniędzy na buty.

Zmniejszenie kosztów budowy komputerów osiągnięto początkowo wprowadzając sztuczne nakazy — żaden program nie mógł zajmować więcej niż „ileś" pamięci. To „ileś" stanowiło więc istotny parametr jakości komputera.

Z czasem jednak, wspomniana wcześniej pierwsza cecha człowieka doszła do głosu. Lenistwo spowodowało, że komputer rozwiązywał coraz bardziej złożone problemy. W konsekwencji, rosły rozmiary programów. W pewnym momencie objętość pamięci operacyjnej okazała się zbyt mała. Jednocześnie układy pamięci były jeszcze na tyle kosztowne, że rozbudowa ich nie wchodziła w rachubę. Postanowiono więc, duży program podzielić na kilka mniejszych części i wykonywać je jedna po drugiej. Programista został jednak obarczony dodatkowym problemem. Pisząc program, musiał zaprogramować rozwiązanie podstawowego zadania i podjąć decyzję, w jaki sposób podzielić program na segmenty. Programista to też człowiek — jest więc leniwy. „Chcemy żyć wygodnie i beztrosko" — krzyknęli programiści. „Nic nas nie obchodzą problemy komputera ze zmieszczeniem programu w pamięci. To jest jego problem, niech więc on sam sobie z tym radzi". Program umieszczono w szybkodostępnej pamięci dyskowej a pamięć operacyjną podzielono na bloki i każdemu dobudowano rejestr adresu. Odtąd każdy blok pamięci miał dwa adresy: adres fizyczny — określający jego położenie w komputerze oraz adres logiczny — według którego procesor mógł w nim zapisywać lub odczytywać dane.

Jednocześnie próba odwołania się do nieistniejącego adresu logicznego powodowała, że procesor otrzymywał sygnał przerwania. Zawieszał wtedy działanie aktualnego zadania, odnajdywał brakujący segment programu na dysku i po przeadresowaniu jednego z bloków, wprowadzał szukany segment do pamięci, po czym uaktywniał zadanie. Program wprawdzie nadal był dzielony na segmenty, jednak teraz odbywało się to bez udziału programisty. Długość każdego segmentu była równa rozmiarowi bloku pamięci operacyjnej. Od tej chwili program mógł być tak duży, jak rozległy był obszar pamięci, który można było zaadresować na szynie adresowej niezależnie od fizycznej wielkości pamięci operacyjnej. Tak zorganizowaną pamięć nazwano pamięcią wirtualną. Koszt rozbudowy szyny adresowej, w porównaniu do kosztu rozbudowy pamięci, jest znikomy.

Z lenistwa i chciwości wyniknęły dwa oszustwa. Pierwsze — pozornie jednoczesne wykonywanie się kilku programów, drugie — pozornie olbrzymia pamięć operacyjna.

Moim zdaniem są to dwa, najpoważniejsze osiągnięcia w dziedzinie rozwoju architektury komputerów.

mgr Mieczysław Płacheta



NOWOŚĆ?

Przebój sezonu 1983 na komputerowym rynku, czyli Commodore 64 dotarł już do Pewexu, wiosną 87 r. Zaskoczenie okazało się wielkie w równym stopniu dla potencjalnych nabywców co dla... krajowego importera, bowiem kilkugodzinne próby uzyskania szczegółowych informacji u źródła, czyli w firmie Pewex nie należały do udanych. Rozmówcy mniej lub bardziej uprzejmie przekazywali wyłącznie numery telefonów różnych działów funkcyjnych i zamiast informacji o mikrokomputerach udzielono nam precyzyjną lekcję poglądową o strukturze organizacyjnej przedsiębiorstwa, wykazując tym samym totalną ignorancję w dziedzinie sprzedawanego przez Pewex asortymentu towarów. W tej sytuacji prawdziwym zaskoczeniem było uzyskanie zwykłej, rzetelnej informacji. Autorem niespodzianki była Spółka z o.o. MERCOMP, prowadząca serwis Commodore, która przejęła po Pewexie działalność promocyjną. Spółka posiadająca tak poważnych współudziałowców jak LOT, POLON czy ERA, przeszkoliła swych pracowników w Wielkiej Brytanii, posiada dokumentację serwisową, niedostępną w naszym kraju i prowadzi serwis gwarancyjny mikrokomputerów na razie tylko zakupionych w Pewexie.

WIEŚCI

Jednopłytkowy koprocesor **dla IMB PC/XT/AT i kompatybilnych wykorzystujący mikroprocesor motoroli 68000 (lub 68010) oferuje firma Quin Systems Ltd. Płyta, wkładana do pełnowymiarowego gniazda (slot) PC, dysponuje mocą obliczeniową minikomputera 16/32-bitowego z zegarem 10 MHz i systemem operacyjnym OS9 wielodostępnym i wielozadaniowym typu UNIX (Bell Laboratories). Możliwe jest również zakupienie systemu operacyjnego DOS68, zapewniającego kompatybilność z systemem DOS IBM PC oraz rozbudowa pamięci 512KB do 1MB. Na płycie przewidziane jest również miejsce na procesor matematyczny 32081, wykonujący 20 000 operacji zmiennoprzecinkowych na sekundę.**

●●●●●

Automatyczny tłumacz **o nazwie FORTRIX firmy DATAWARE przekształca w otoczeniu UNIX/C pliki i programy z FORTRAN-u na język C z szybkością 600 linii na minutę. Program umożliwia również naukę programowania w języku C programistom znającym FORTRAN.**

●●●●●

Wielostandardowy monitor **kolorowy, kompatybilny z kartami CGA,EGA i PGA IBM PC oferuje firma brytyjska Reflex Ltd. Monitor może również współpracować z innymi mikrokomputerami z częstotliwością odchylania poziomego w zakresie od 15 do 34 kHz. Po włożeniu i zainstalowaniu dowolnej z wymienionych kart monitor automatycznie dostosowywuje się do właściwego trybu pracy, akceptując sterowanie sygnałami analogowymi lub TTL.**

●●●●●

80C31 **jest 8-bitowym mikroprocesorem wykonanym w technologii NMOS w firmie Advanced Micro Device, co w stosunku do jego odpowiednika w technologii NMOS pozwala na zmniejszenie poboru mocy o 75%. W tej samej kości znajduje się również 128KB pamięci RAM, 32 programowalne wejścia/wyjścia, dwa liczniki 16-bitowe, wbudowane przerwania z pięciu źródeł i dwóch poziomach, dwukierunkowa brama szeregowa, generator wewnętrzny i zegar. Układ wykonany jest w 40-nóżkowej obudowie typu DIL, przystosowanej do montażu powierzchniowego.**

●●●●●

wz

Drogi Bajtku!



Na listy czytelników odpowiada Marcin Waligórski.

Wymyśliłem prosty program, przy pomocy którego komputer sprawdza, czy dana liczba jest pierwsza, czy złożona. Oto on:

```
10 INPUT „L="; L
20 IF L < 2 OR L > 32767 OR L <> INT (L) THEN GOTO 10
30 FOR N=1 TO L
40 LET K = L / N
59 IF K = INT (K) AND K <> 1 AND K <> L THEN
PRINT „LICZBA ZŁOŻONA": GOTO 80
60 NEXT N
70 PRINT „LICZBA PIERWSZA"
80 STOP
```

Wprawdzie program ten działa dobrze, ale przy większych liczbach czas oczekiwania na sprawdzenie jest bardzo długi. Czy moglibyście zaproponować inny algorytm rozwiązania tego problemu?

Jerzy R.
(nazwisko i adres do wiad. redakcji)

Program jest faktycznie prosty, co nie znaczy, że użytego w nim algorytmu nie można usprawnić. Zauważmy, że dla liczby pierwszej L algorytm wykonuje dokładnie L dzieleń i sprawdzeń wyniku. Na pierwszy rzut oka widać, że wystarczy tylko L-2 dzieleń, bo nie musimy sprawdzać podzielności przez 1 i przez L. Wystarczy zmienić linię 30 programu na

30 FOR N=2 TO L-1

i nieco uprościć sprawdzanie warunków w linii 50.

Przy dużych L oszczędność dwóch dzieleń nie ma oczywiście większego znaczenia. Poszukajmy zatem większych oszczędności. Zobaczmy, że liczba podzielona przez dowolne N parzyste dzieli się również przez 2. Wystarczy zatem najpierw zbadać podzielność przez 2, a potem możemy już ograniczyć się do nieparzystych N, tzn. 3,5,7,9... Modyfikacja jest prosta, a zmniejszy ilość wykonywanych działań o mniej więcej połowę — do L/2.

Jak powiada Dennie van Tassel: „Wielu ludzi sądzi, że nie można już niczego poprawić, bo już raz poprawialiśmy". A jednak. Przyjrzyjmy się jeszcze raz liczbie L. Jeżeli jest ona złożona, to da się przedstawić w postaci iloczynu

$$L = N * K$$

gdzie N i K są oczywiście podzielnikami liczby L. Zobaczmy, że jeden z tych dwóch podzielników jest zawsze mniejszy bądź równy pierwiastkowi z L.

$$N \leq \sqrt{L} \text{ lub } K \leq \sqrt{L}$$

Łatwo to udowodnić i czytelnik z pewnością dokona tego we własnym zakresie. Wniosek dla nas z tego wypływający jest taki, że jeżeli liczba posiada podzielnik większy od $\sqrt{L}$, to na pewno posiada podzielnik mniejszy bądź równy $\sqrt{L}$. To pozwala ograniczyć zakres sprawdzanych N z góry jedynie do pierwiastka z L. Ograniczamy zatem znów liczbę działań do wielkości $\sqrt{L/2}$.

Zobaczymy, jaka to oszczędność: dla L rzędu 10 000 wykonujemy w tym ostatnim przypadku 50 (pięćdziesiąt!) dzieleń wobec dziesięciu tysięcy w pierwszej wersji programu. Znów cytując van Tassela: „Godzina planowania warta jest pięciu godzin programowania".

Oto poprawiona wersja programu:

```
10 INPUT „L="; L
20 IF L < 2 OR L > 32767 OR L <> INT (L) THEN
GOTO 10
30 LET K = L / 2: REM Sprawdzamy podzielność
przez 2
40 IF K=INT (K) THEN GOTO 120
50 LET Zakres = SQR (L)
60 FOR N=3 TO Zakres STEP 2
70 LET K = L / N
80 IF K=INT (K) THEN GOTO 120
90 NEXT N
100 PRINT „LICZBA PIERWSZA"
110 STOP
120 PRINT „LICZBA ZŁOŻONA"
130 STOP
```

Błędne byłoby przekonanie, że ilości działań nie da się jeszcze zmniejszyć. Gdybyśmy np. znali wszystkie liczby pierwsze mniejsze od $\sqrt{L}$, wystarczyłoby sprawdzać podzielność tylko dla nich. Dlatego też algorytm znajdowania wszystkich liczb pierwszych od 1 do L rozwiązalibyśmy całkiem inaczej. Prostą, acz wymagającą stosunkowo dużo pamięci metodą byłoby wówczas również tzw. sito Eratostenesa, który to algorytm w ogóle nie wymaga wykonywania innych działań niż dodawanie.

Odsyłam do literatury: Niklaus Wirth „Wstęp do oprogramowania systematycznego", WNT 1987 oraz cytowanej tu publikacji Dennie van Tassela „Praktyka programowania", WNT 1978. Obie książki zostały wydane w serii BIO.

Nie mam wprawdzie własnego komputera, ale w szkole znajduje się ZX SPECTRUM +. Proszę o pomoc w następujących problemach: Chcę, aby podczas wykonywania programu pewien napis był cały czas widoczny. Czy można coś zrobić, aby instrukcja CLS kasowała wszystko oprócz tego napisu?

(nazwisko i adres do wiadomości redakcji)

Szybkie kasowanie wybranych części ekranu jest możliwe do zrealizowania przy użyciu kodu maszynowego. Trudno tu modyfikować systemową procedurę kasowania ekranu — prościej jest napisać własną. Napisanie takiej procedury, kasującej wszystko oprócz wybranego „okienka" jest w ogólnym przypadku trudne. Można zadanie uprościć, wykorzystując fakt podziału pamięci ekranu na trzy bloki po 2 kilobajty (jak te bloki wyglądają na ekranie można zobaczyć podczas wczytywania obrazka z magnetofonu). Jeżeli chcemy skasować wybrany blok ekranu, wystarczy wyzerować 2 kB pamięci, począwszy od adresu:

16384 — dla górnego bloku,
18432 — dla bloku środkowego,
20480 — dla dolnego bloku ekranu.

Kasowania dużych obszarów pamięci najłatwiej dokonać, stosując pewien sposobik z użyciem instrukcji LDIR procesora. Oto on, przedstawiony schematycznie:

```
LD (Adres), 0
LD HL, Adres + 1
LD DE, Adres
LD BC, Długość - 1
LDIR
```

Adres jest tu wielkością wyznaczającą początek kasowanego bloku, zaś Długość — ilością bajtów do wyzerowania. Podczas wykonania instrukcji LDIR zero spod adresu Adres zostanie skopiowane pod Adres + 1. Następnie kopiowana jest komórka Adres + 1 do komórki Adres + 2 — ale przecież przed chwilą do Adres + 1 wstawione zostało zero, zatem komórka Adres + 2 zostaje wyzerowana, i tak dalej. Jak widać, cały program liczy tylko kilka bajtów. Jak już wspominałem, rozwiązanie bez przyjętych założeń upraszczających byłoby znacznie bardziej skomplikowane.

Jak zrealizować dzielenie lub mnożenie w asemblerze?

Jak zrealizować przesunięcie kilku bajtów o jeden bit w lewo lub w prawo? (dotyczy procesora Z80 — przyp. MW).

Czy program studiów informatycznych obejmuje języki programowania wysokiego rzędu, czy asemblery?

W jaki sposób realizowane jest w grach sprawdzenie, czy zaszło spotkanie np. bohatera z duszkiem itp.?

Bogdan Piguła
uczeń LO im. Małachowskiego
Płock

Asemblery różnych typów najczęściej udostępniają programiście instrukcje dodawania i odejmowania plus zestaw arytmetycznych operacji bitowych (np. przesunięcia, negacje). Z tego powodu realizacja działań mnożenia i dzielenia może być w ogólnym przypadku nieco kłopotliwa. Łatwo zauważyć jedynie to, że przesunięcie reprezentacji liczby o jeden bit w lewo (przy wstawieniu bitu 0 na pierwsze miejsce) odpowiada przemnożeniu liczby przez 2. Odwrotnie, przesunięcie w prawo — znów ze wstawieniem najstarszego bitu 0 — odpowiada dzieleniu całkowitemu liczby przez 2. Ale uwaga — powyższe reguły dotyczą arytmetyki bez znaku. Dla arytmetyki ze znakiem trzeba dodatkowo dbać o zachowanie najstarszego bitu, który jest właśnie bitem znaku. Najczęściej procesor może wykonywać dwa rodzaje przesunięć: arytmetyczne i logiczne, odpowiadające tym dwom sytuacjom.

Podprogram mnożenia liczb można utworzyć, wykorzystując właśnie przesunięcia i dodawanie. Podobnie rzecz ma się z dzieleniem.

Prościej jednak sięgnąć do gotowych procedur arytmetycznych, umieszczonych w systemie operacyjnym komputera. Większość systemów operacyjnych jest wyposażona właśnie w tzw. arytmetykę systemową, stanowiącą właśnie zbiór procedur obliczających nie tylko np. iloczyn i iloraz, lecz także inne funkcje — choćby pierwiastek kwadratowy lub sinus.

Przesunięcia całych grup bajtów (niezbędne przy realizacji działań na liczbach podwyższonej precyzji) najczęściej można dokonać przy użyciu wskaźnika (bitu) C-Carry procesora. Przy wykonywaniu przesunięcia pewnego bajtu, bit „gubiony" — najstarszy przy przesunięciu w lewo i najmłodszy przy przesunięciu w prawo — zostaje skopiowany do C. Podczas przesuwania następnego bajtu możemy C wstawić z powrotem — z tym, że na nowym już miejscu. Procesor Z 80 posiada zresztą w zestawie instrukcji przesunięcia ze wstawieniem Carry — są to instrukcje RLA i RRA oraz RL i RR.

Jako przykład programu studiów wyższych na kierunku informatyka mogę podać program Informatyki UW. Obejmuje on kilka języków programowania wyższego rzędu (jak na razie z PASCAL-em na pierwszym miejscu) oraz jedną całoroczną pracownię programowania w asemblerze. Warto jednak zauważyć, że główny nacisk program ów kładzie na poznanie informatycznych metod rozwiązywania problemów i tworzenia oprogramowania, a nie na poznawanie coraz to nowych języków.

Sprawdzania kolizji obiektów na ekranie najczęściej dokonuje się po prostu przez porównanie ich współrzędnych lub obliczenie wzajemnej odległości. Oba podejścia wymagają pamiętania w każdym momencie współrzędnych wszystkich obiektów. Nie muszą to być współrzędne ekranowe, ale np. miejsca obiektów w pewnej tablicy, odwzorowującej ekran.

W niektórych mikrokomputerach (np. Atari XL) kolizje obiektów są sygnalizowane przez procesor graficzny. Oczywiście znakomicie ułatwia to pracę programiście.

Proszę o wymienienie tytułów symulatorów lotu myśliwcem na ZX Spectrum.
(nazwisko i adres do wiadomości redakcji)

W tej chwili najlepszym — i chyba jedynym dobrym — programem tego typu na Spectrum jest „Fighter Pilot", którego autorzy powołują się na pierwowzór w postaci samologu F-15 Eagle. Ze względu na kiepską grafikę Spectrum symulatorowi temu daleko jeszcze do doskonałości. Osobiście uważam, że więcej satysfakcji daje nocny lot Lancasterem w grze „Dambusters", choć to śmigłowy samolot bombowy, zaś program nie ma pretensji do nazwy „symulator".

Marcin

Od niedawna używam kompilatora PASCAL-a firmy Hisoft HP4T1.6M na komputerze ZX Spectrum 48K. Kompilator ten w instrukcji READ nie przyjmuje danych typu CHAR, chociaż program nie wykazuje błędów. Zgodne jest to z uwagą p. Marka Wyrwidęba zamieszczoną w artykule z nr-u 12/86. Rozwiązać ten problem można za pomocą zdefiniowania dwóch procedur:

```
PROCEDURE WPISZ;
BEGIN
  ZNAK := INCH;
  WHILE (ZNAK<CHR (32)) OR (ZNAK > CHR (127))
  DO
  ZNAK := INCH
END;
PROCEDURE PAUSE;
VAR I: 1..5000;
BEGIN
  FOR I := 1 TO 5000 DO
  BEGIN
  END
END;
```

Funkcja INCH działa analogicznie jak INKEY$ w Basic-u, tzn. jej wartością jest znak reprezentowany przez ostatnio wciśnięty klawisz. Oczywiście w programie bezpośrednio przed procedurą WPISZ należy umieścić procedurę PAUSE.
Mam jednak pewne wątpliwości co do mojego spostrzeżenia dotyczącego procedury READ. Wątpliwości te powoduje wypowiedź p. Kuryłowicza w książce „Przewodnik po ZX Spectrum", s. 213. Cytuję:
„...próba wczytania liczby lub łańcucha znaków spowoduje błąd. Dlatego pierwszą instrukcją czytania powinno być READLN".
Pan Kuryłowicz opisuje kompilator HP4T1.4, czyli poprzednią wersję HP4T1.6M.
Jeżeli p. Markowi Wyrwidębowi wiadomo coś bliższego na ten temat, prosiłbym w miarę możliwości o odpowiedź.

Marek Sitnicki
ul. Bratysławska 5a m. 75
94-032 Łódź

Oto odpowiedź autora „Wstępu do programowania w języku PASCAL" opublikowanego w nr-ach 10/86—1/87 „Bajtka": „Wstęp do programownia w PASCAL-u — jako że był tylko wstępem — nie zawarł w ogóle opisu standardowej procedury READLN. Jej wywołanie powoduje „przejście z czytaniem do nowej linii" — analogicznie jak w parze WRITE i WRITELN. Dosłownie oznacza to pominięcie wszystkich znaków z wejścia aż do napotkania znaku końca (CHR(13)), a następnie dodatkowo pominięcie tego znaku, aby następny wczytany znak był pierwszym znakiem nowej linii. w PASCAL-u Hisoft, nie wnikając w szczegóły implementacyjne, możemy akcję taką interpretować jako opróżnienie bufora wejściowego, zawierającego ostatnio wczytaną linię. Po wywołaniu READLN następna instrukcja READ spowoduje ponowne wypełnienie bufora wejścia tym, co użytkownik wypisze na klawiaturze, a następnie odczytanie z bufora potrzebnych danych.

Jak widać, sam proces czytania wykorzystuje dane zawarte w buforze, a nie pobiera ich na bieżąco z klawiatury. Kłopot w posiadanej przez Pana wersji języka jest taki, że zaraz po uruchomieniu bufor wejściowy zawiera linię pustą — zatem tylko znak CHR (13). Próba czytania tego znaku to oczywiście odczytanie znaku „karetki" i nic więcej — zatem błędu nie ma, są tylko wczytane błędne dane.

Użycie READLN przed READ rozwiązuje problem.

Proszę zauważyć, że takich kłopotów nie ma, gdy wczytujemy liczby — wtedy PASCAL automatycznie pomija wszystkie znaki nie będące cyframi, +, − lub E.

Pański program stanowi ominięcie naszego problemu — po prostu zrezygnował Pan z usług bufora i zajął się czytaniem bezpośrednio z klawiatury.

Na zakończenie tych kilku uwag pragnę zauważyć, że operacje wejścia/wyjścia są tym miejscem, gdzie najczęściej implementacja różni się od raportu języka. Jest to też potencjalne miejsce różnic między różnymi kompilatorami.

Z poważaniem
Marek Wyrwidąb

Pan M. Kop w swym liście („Bajtek" 2/87) pomawia Studio „Jacke" o rozbój, zapewne w trosce o swe bezpieczeństwo nie ujawniając adresu.

Zabieram głos nie tyle w obronie własnej czci, co dla podyskutowania o szerszym problemie ukazanych w liście p. M. Kop postaw moralnych i wychowawczych oraz sposobu pojmowania zasad działania rynku oprogramowania i literatury komputerowej.

Pan M. Kop pisze: „syn jest stałym bywalcem giełdy". Bywalcem czy handlarzem? „Giełda jest wspaniałym rozwiązaniem chroniącym moją kieszeń". Dziękuję za odpowiedź. A może pociecha zamiast na giełdzie pracować na ojca zajęłaby się własną nauką lub pracą, zwłaszcza że na giełdach ponoć biją?

Co jednak dziecku przeszkadza zarabiać na papu? Drobnostka: z braku własnego dorobku ma ono do sprzedania jedynie cudzy, trzeba więc okradanym zabronić chronienia swych praw.

Pan Kop proponuje, a Bajtek bez zastrzeżeń propaguje podwójną moralność: okraść Kalego (swoich) źle, okraść obcych — dobrze. Jedyny problem w tym, że okradani coraz gorzej pracują i poziom ich opracowań „woła o pomstę do nieba".

Właśnie by zapewnić wysoki poziom literatury, instrukcji oraz — zazwyczaj niezwykle pracochłonnych — adaptacji oprogramowania nieliczne prywatne firmy (m.in. „Pro-Info" i „Studio Jacke") zatrudniają czołowych polskich informatyków, popularyzatorów, płacąc im za wysokiej jakości i w ekspresowych terminach wykonywaną pracę kwoty przekraczające nieraz milion za pojedynczy tytuł, dzięki czemu podstawowe informacje techniczne o najnowszych światowych osiągnięciach z zakresu konstrukcji sprzętu i oprogramowania dostępne są zarówno na poziomie profesjonalnym, jak i popularnym nie po 5 latach, jak w księgarniach, a po 2—3 miesiącach. Korzysta na tym cała gospodarka i kultura narodowa.

Aby ponoszenie tych nakładów było możliwe prace te trzeba następnie sprzedać w dostatecznej ilości i niestety po odpowiedniej cenie (choć i tak znacznie niższej niż życzą sobie za podobne opracowania nasze instytuty naukowe). Pan Kop woli zbierać nie siejąc i uważa, że jedyne liczące się koszty to koszty kserografu, na którym jego syn powiela nielegalnie cudze prace.

W ciągu minionych lat zamówiłem i wydałem wiele podręczników i programów cieszących się uznaniem użytkowników Atari, Commodore, Amstrada, a ostatnio IBM, np. całkowicie oryginalne podręczniki do polskiej wersji Framework II (cztery tomy), GW Basic (dwa tomy), programów Drukarz (Lettrix), Sidekick, Turbo-Pascal, Smartwork, MSWord, Turbo-Prolog, Chwriter autorstwa m.in. R. Wacławka, J. Orkiszewskiego, W. Majewskiego, L. Rudaka i innych.

Janusz Gołuch
Studio „Jacke"

WSZYSTKO DLA WSZYSTKICH



ZAKŁADY URZĄDZEŃ KOMPUTEROWYCH

MERA-ELZAB

41-813 Zabrze, Kruczkowskiego 39
telex 036711 telefon 72-20-21 do 29

Oferują do sprzedaży mikrokomputer

COMPAN 8/16

Z NATYCHMIASTOWĄ DOSTAWĄ

COMPAN 8/16 to:

- duża pamięć operacyjna, w tym RAM DYSK o pojemności 448 kB
- pamięć masowa na dyskach elastycznych 5.25" 2x720 kB i/lub na dysku twardym typu WINCHESTER o pojemności 22 MB
- grafika o dużej rozdzielności 288x640
- klawiatura z klawiszami programowalnymi w układzie zgodnym z mikrokomputerami XT
- możliwość pracy w wielodostępie
- konkurencyjne ceny w złotówkach

SKORZYSTAJ Z OKAZJI!!!
NA ŻYCZENIE WYSYŁAMY AKTUALNĄ OFERTĘ.

k-128

EMD HONG KONG
KOMPUTERY

PC-1011 (komp. PC/XT) ..... US $ 690,-
PC-1021 (komp. PC/AT) ..... US $ 1499,-
plus 20 MB HDD i inne liczne opcje.

- ceny z dostawą do Polski
- roczną gwarancją i serwis w Polsce

Szczegółowe informacje, cenniki itp. do uzyskania w biurze sprzedaży EMD w Wiedniu:

EMD (HK) Warenhandelsges. m.b.H.
Postfach 214
A-1041 Wien, Austria.

K-95

Wojewódzkie Przedsiębiorstwo Handlu Wewnętrznego
Oddział w Tychach

„VIDEOBIT"

43-100 TYCHY, ul. Al. ZMP 77, tel. 27-69-75

PROWADZI
SKUP — SPRZEDAŻ

— mikrokomputerów
— urządzeń peryferyjnych
— oprogramowania
— sprzętu magnetowidowego

Zapewniamy o atrakcyjnych cenach.
Sklep prowadzi sprzedaż pozarynkową.

K-106

COMPUTRONIX
PSS SPOŁEM

Kraków, Garnearska 21
oferuje

- **sprzęt komputerowy wraz z peryferiami**

SKLEP WYDAJE RACHUNKI.

K-110

PROGRAMY DO C 64/128
ATARI ST, AMIGA, IBM, PCW
Sławomir Krysztofowicz, ul. Iwaszkiewisza 1/71, 42-200 Częstochowa.

G-19

ASTRO-KOMPUTER STUDIO
PROGRAMY
● ATARI ● SHARP MZ-700, 800 ● ZX SPEKTRUM
w sprzedaży wysyłkowej
54-515 WROCŁAW,
ul. GDACJUSZA 39

G-58

ZX SPECTRUM — programy — wymiana.
Andrzej Hofman, ul. I Armii W.P. 4 m. 41, 43-300 Bielsko-Biała.

G-59

PUSZKA PANDORY

ENTER computing

Autor:
M. BORKOWSKI

- **pierwsza polska pełnowymiarowa graficzno-tekstowa gra typu adventura dla ZX SPECTRUM**
- **zadaniem grającego — odnaleźć i unieszkodliwić system atomowego zagrożenia.**

ENTER-COMPUTING 02-105 WARSZAWA P-3
INF. PO NADESŁANIU KOPERTY ZWROTNEJ.

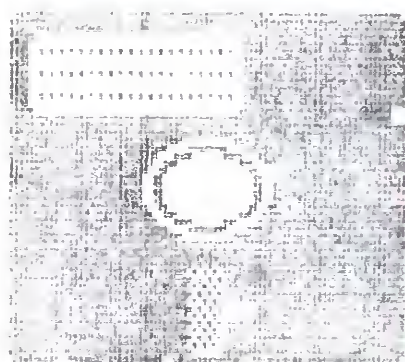
D-70

INTERFEJS CRI

wykonywany na zamówienie przez firmę IBS-elektronic umożliwia współpracę zwykłego magnetofonu z komputerem **ATARI** zapis, odczyt, start-stop magnetofonu gwarancja
Warszawa, tel. 34-16-06 w g. 10.00—14.00.

D-100

Redakcja nie ponosi odpowiedzialności za treść ogłoszeń.



INDYWIDUALNY BANK DANYCH

Nazywam się **Waldemar Stós**. Jestem studentem medycyny, mam 21 lat. Posiadam mikrokomputer Commodore VIC 20. Oprogramowanie: gry zaczerpnięte z czasopism o Commodore i z oryginalnych książek zawierających programy na ten mikrokomputer; programy firmowe- Sargon II, Road Race, Alpha Alarm. Interesuję się muzyką, uprawianiem takich sportów, jak tenis ziemny i wycieczki rowerowe. Proponuję wymianę software'u, oferuję sporo gier rozrywkowych i programów muzycznych. Mój adres: ul. Obrońców Stalingradu 1, 32-800 Brzesko, woj. tarnowskie.

Klaudiusz Urbański, uczeń 12 lat. Mikrokomputer Atari 130XE i magnetofon. Oprogramowanie: użytkowe i gry. Proponuję wymianę informacji na temat użytkowania Atari 130XE oraz programów. Zainteresowania: elektronika, muzyka. Adres: ul. Batorego 38 m. 23, 05-400 Otwock.

Michał Malordy, uczeń, 14 lat. Mikrokomputer ZX Spectrum + oraz magnetofon. Oprogramowanie: programy użytkowe, graficzne, gry, podstawowe elementy BASI-Ca. Zainteresowania: elektronika, informatyka, filatelistyka. Adres: ul. Bytomska 13, 41-600 Świętochłowice.

Ryszard Krakowiak, programista, 40 lat. Mikrokomputer Sharp MZ 731 z drukarką. Oprogramowanie: edytor tekstu, kartoteka, adresy, biorytmy, kompilator BASIC, kompilator PASCAL-a, Assembler, szachy oraz gry w wersji polskiej. Zainteresowania — astrologia. Adres: ul. Gdacjusza 39, 54-515 Wrocław.

Artur Chromy, uczeń, 16 lat. Mikrokomputer Commodore 116 z magnetofonem. Oprogramowanie: programy graficzne, muzyczne, Turbocopy, firmowe-szachy, brydż, Wizard, Major Blink i inne gry. Zainteresowania: fizyka, matematyka, elektronika. Proponuję wymianę literatury, własnych rozwiązań oraz gier. Adres: Czyżyńska 2/1, 31-571 Kraków. Szczególnie zależy mi na kontakcie z „mikrokomputerowcami" z Krakowa, ponieważ jestem niepełnosprawny.

Edyta Malec, uczennica, 16 lat. Mikrokomputer Meritum I.Software — głównie gry. Zainteresowania: geografia, piłka nożna. Oczekuję wymiany gier. Adres: ul. Wronia 7/5, 59-300 Lubin, woj. legnickie.

Krzysztof Przyłucki, uczeń, 17 lat. Mikrokomputer ZX Spectrum 128 oraz magnetofon firmowy Dicsons. Oprogramowanie: edukacyjne — kombinatoryka, wyznaczniki, demonstracyjne — książka telefoniczna i bank adresów, gry, ilustracja prawa wielkich liczb Bernoulliego, programy kopiujące, translatory — Pol/Logo, Beta- Basic. Zainteresowania: matematyka, psychologia, języki obce. Proponuję wymianę oprogramowania i doświadczeń. Adres: Gawliki Wielkie 30, 11-510 Wydminy.

Rafał Konopacki, uczeń. Mikrokomputer MSX-Sony HIT BIT. Z powodu nietypowości mojego mikrokomputera pilnie poszukuję oprogramowania i literatury na jego temat. Adres: ul. Łączności 6a/2, 53-330 Wrocław.

Radosław Szatkowski, uczeń, 13 lat. Mikrokomputer Amstrad CPC 464. Oprogramowanie: programy edukacyjne, graficzne, obliczanie pola koła, obliczanie gęstości masy, plan lekcji. Chciałbym nawiązać kontakt z posiadaczami tego typu komputera celem wymiany doświadczeń, programów i literatury. Adres: ul. Gwardii Ludowej 4b/8, 26-200 Końskie.

Jerzy Wiśniewski, chemik, 37 lat. Mikrokomputer Atari 800XL. Oprogramowanie: gry i kilka programów użytkowych (Logo, Assembler, Translator). Proponuję nawiązanie kontaktu w celu wymiany programów i współpracy w zakresie programowania. Adres: ul. Śląska 15/2, 85-235 Bydgoszcz.

Przymysław Lewkonowicz, student, 20 lat. Mikrokomputer Oric-1. Oprogramowanie: obliczanie wytrzymałości belek zginanych, nauka alfabetu Morse'a, program demonstracyjny Oric Software. Zainteresowania: elektronika, informatyka. Adres: ul. Bratnia 1, 56-400 Oleśnica Śl.

Tomasz Wiese, uczeń, 15 lat. Mikrokomputer Atari 65XE i magnetofon XC-12. Oprogramowanie firmowe (gry, programy użytkowe i edukacyjne). Zainteresowania: matematyka, fizyka, elektronika, muzyka. Wymiana oprogramowania i literatury. Adres: ul. II Armii WP 17, 64-610 Rogoźno Wlkp.

KLUB MIKROKOMPUTEROWY C.K. ATARI

Klub C.K.M. ATARI przy choszczeńskim Domu Kultury powstał na początku 1986 r. Obecnie dysponujemy 3 klubowymi komputerami ATARI 800 XL z firmowymi magnetofonami, ponadto stacją dysków ATARI 1050 i drukarką ATARI, czyli systemem mikrokomputerowym. Jesteśmy w posiadaniu ok. 200 programów na mikrokomputer ATARI. Wymieniony sprzęt został zakupiony przez choszczeński Dom Kultury, a także z dotacji min. ds. Młodzieży tow. Aleksandra Kwaśniewskiego, który wizytował Klub i jego działalność w 1986 r.

Zasady działania klubu

Klub mikrokomputerowy C.K.M. ATARI postawił sobie za zadanie:

— programowanie wiedzy ogólnej o zastosowaniach mikrokomputerów i systemów mikrokomputerowych

— skupienie tych, którzy wykazali szczególne zainteresowanie tematem

— indywidualnym posiadaczom mikrokomputera ATARI służymy pomocą,

— członkiem klubu może zostać każdy, kto aktywnie uczestniczy w działalności klubu.

— wszystkie publikacje i programy będące w posiadaniu klubu i jego członków są udostępniane bezpłatnie.

Klub i jego działalność

Klub prowadzi działalność w kierunkach:

1. Propagowanie ogólnej wiedzy mikrokomputerowej w społeczeństwie.
2. Pokazy konkretnych zastosowań techniki mikrokomputerowej.
3. Działalność wewnątrz klubową polegającą na:

— przyswojeniu umiejętności obsługi, wykorzystania i zastosowań systemów mikrokomputerowych jak i programów użytkowych i narzędziowych napisanych przez profesjonalistów

— samodzielne programowanie i praca ze sprzętem.

W naszym regionie jesteśmy miejscem spotkań dla fanów z okolicznych miejscowości takich jak: Barlinek, Stargard Szczeciński, Gorzów i innych. W naszym klubie powstało kilka programów użytkowych m.in. adresowa baza danych, kosztorys robót malarsko-budowlanych, obliczający zużycie materiałów, roboczogodziny i koszt całkowity prac, oraz program obliczający ceny uśrednione. Pomocą służy nam klub ATARI ze Szczecina liczący ok. 200 członków.

— Zamierzenia i plany na rok 1987

Kontynuacja realizowanego planu. Zakupić dalsze mikrokomputery, ponieważ tłok nie sprzyja pracy, napisanie skryptu dla początkujących, podejmowanie działań w kierunku zwalczania zjawisk patologicznych poprzez rozbudzanie zainteresowań i zagospodarowywanie czasu wolnego.

Dyrektor
mgr Jan Kiela

„SHARP" NA START!

Klub Użytkowników Komputerów Osobistych „MIKRO-KOMP" w Łodzi, działający w ramach Ogólnopolskiej Federacji Klubów Komputerowych przy Radzie Naczelnej Zrzeszenia Studentów Polskich informuje, że w dniach 17—18 października 1987 r. w Łodzi odbędzie się I Zjazd Użytkowników Komputerów „SHARP".

Miejsce Zjazdu: Dzielnicowy Dom Kultury Łódź-Polesie, ul. 1 Maja 87. Celem Zjazdu jest konsolidacja środowiska, wymiana informacji i doświadczeń, przedstawienie własnych osiągnięć członków sekcji SHARP, popularyzacja nowych rozwiązań sprzętowych i programowych. W programie — wykłady i referaty dotyczące użytkowania komputera SHARP MZ.

Do wzięcia udziału w Zjeździe zapraszamy również użytkowników nietypowych komputerów z mikroprocesorem Z-80, gdyż ich też powinna zainteresować tematyka poruszanych zagadnień.

Istniejąca od 1985 r., jedyna w kraju sekcja użytkowników SHARP-a, oraz również jedyna w kraju sekcja użytkowników komputerów nietypowych z procesorem Z-80, działające w ramach naszego Klubu, zrzeszają w swych szeregach posiadaczy takich komputerów z terenu całej Polski.

Informacje na temat Zjazdu można uzyskać w Biurze Organizatora: D.D.K. Łódź-Polesie ul. 1 Maja 87, tel. 33-08-02 lub bezpośrednio u osób koordynujących przygotowania do Zjazdu: V-ce prezes Klubu, szef sekcji SHARP — GARLICKI JERZY ul. A. Struga 4, tel. 32-57-83
V-ce prezes Klubu, szef sekcji NIETYPOWYCH — ŻYŁŁA ROMUALD ul. Zachodnia 12, tel. 57-75-06.

FANCOMCLUB AMIGA

Wszystkich posiadaczy oraz użytkowników komputerów AMIGA zaprasza do współpracy nowo powstały korespondencyjny „Fancomclub AMIGA" w Poznaniu. Przypuszczalnie w kraju jest jeszcze niewielka ilość tych wspaniałych komputerów i do tego znajdują się one w różnych odległych od siebie miastach. Dlatego też celowym jest skupienie „amigowców" w jedną (na razie nieformalną) grupę w celu szerszej i sprawniejszej wymiany doświadczeń, programów i literatury. Klub nasz chciałby pełnić rolę konsolidującą i pośredniczącą, a w dalszych zamierzeniach stworzyć sieć modemową łączącą członków klubu.

Korespondencję wraz z informacją o sprzęcie (wersja DOS-u, RAM, peryferia), dane personalne, oraz wszelkie uwagi i propozycje proszę kierować na adres:

61-688 POZNAŃ
O. Przyjaźni 14/153
Marek Frąckowiak



WAGA

Cześć maluchy!

Macie młodsze rodzeństwo? Jeśli nawet nie macie, to pewnie znajdziecie wśród swoich kolegów takich, którzy rozpoczynają naukę liczenia. Właśnie dla nich przeznaczony jest ten program.

Czytaliście zapewne wiele razy — nie tylko w „Bajtku" — o programach dydaktycznych. Bywają one bardzo rozmaite, ale mają jedną wspólną cechę — korzystając z nich, czegoś się uczymy. A jeśli w dodatku program jest zabawny, to tym lepiej. Nasz program będzie służył do ćwiczenia się w umiejętności liczenia, konkretnie odejmowania na poziomie przedszkolaków (tych prawdziwych, a nie komputerowych). Myślę, że jest on nawet trochę zabawny.

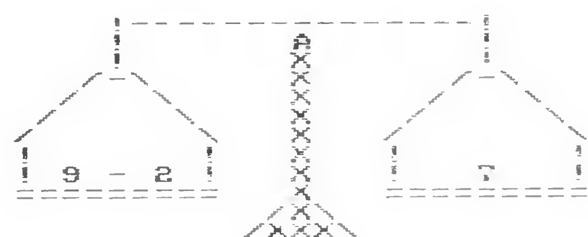
Czy można zważyć liczbę? Chyba raczej nie bardzo, bo czy ktoś miał w ręce np. siódemkę albo dziewiątkę... Jak więc takie coś położyć na wadze? Pofantazjujmy: jeśli by się jednak dało ważyć liczby, to dziewiątka powinna być cięższa od siódemki, a dwie dwójki powinny mieć ciężar jednej czwórki. Jeśli od ósemki odejmiemy piątkę to różnica powinna mieć wagę trójki. Śmieszne ale... logiczne. Prawda?

Taki właśnie niecodzienny pomysł został wykorzystany w naszym programie. Najpierw komputer losuje dwie liczby (zwróćcie uwagę na to, że druga liczba musi być mniejsza od pierwszej) a następnie zadaje pytanie. W zależności od tego, czy odpowiedź jest prawidłowa komputer przechodzi do wykonania odpowiedniego podprogramu rysującego wagę. Na jednej z jej szalek znajduje się zadane do rozwiązania wyrażenie, na drugiej podana odpowiedź. Jeśli jest ona prawidłowa, waga pozostaje w równowadze, jeśli podana liczba jest zbyt duża waga przechyla się na jej stronę, jeśli jest zbyt mała, szalka z odpowiedzią unosi się do góry.

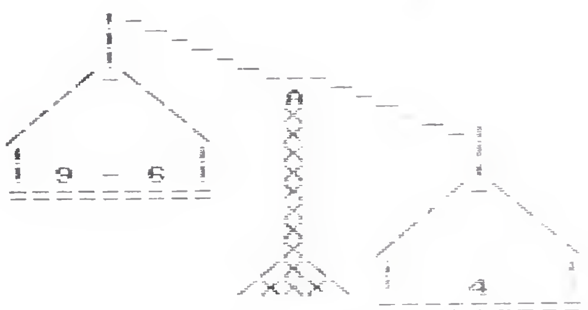
Popatrzcie w jaki prosty sposób skonstruowane zostały podprogramy rysujące wagę. Jedynym problemem mogą tu być linie, w których obok znaków są wyświetlane na ekranie wartości zmiennych. W tym przypadku było to proste, gdyż wszystkie liczby, których używamy mieszczą się w zakresie od 1 do 9, a więc są jednocyfrowe. Jeżeli zdecydujecie się przerabiać ten program i stosować inne działania arytmetyczne, staniecie przed problemem jak wpisać liczby o różnej długości w taki sposób, by nie wpływało to na ustawienie znaków za liczbą. Można to zrobić na bardzo wiele sposobów. Najprostszym jest użycie rozkazu TAB, który ustawia kursor na zadanym miejscu. Możemy również użyć specyficznych dla danego komputera rozkazów umieszczających kursor w dowolnej pozycji ekranu. Dla Spectrum jest to AT, dla Atari — POSITION, dla Amstrada — LOCATE. W Commodore możemy skorzystać ze znaczków sterujących kursorem spod rozkazu PRINT.

Spróbujcie więc rozwinąć ten program, temat jest wdzięczny, bo praktycznie nie stawia żadnych ograniczeń. Możecie dopisać procedury dźwiękowe, ulepszyć grafikę, wzbogacić komentarze i co tylko przyjdzie wam do głowy. Potraktujcie to jako zadanie domowe.

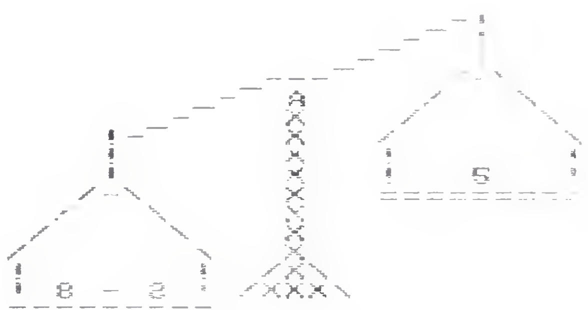
Romek



Bardzo dobrze Kubuś.



Błąd Kubuś! Za dużo.



Błąd Kubuś! Za mało.

```
ATARI 800 XL
1 DIM IMIE$(15)
100 FRINT CHR$(125):
170   LET A=INT(RND(0)*7+3)
180   LET B=INT(RND(0)*8+1)
200   FRINT CHR$(125):
240   FRINT CHR$(125);
320 FRINT CHR$(125):

COMMODORE 64, VC 20
100 FRINT CHR$(147):
200   PRINT CHR$(147);
240   FRINT CHR$(147):
320 FRINT CHR$(147);

SPECTRUM
170   LET a=INT(RND*7+3)
180   LET b=INT(RND*8+1)

MERITUM
170   LET a=INT(RND(0)*7+3)
180   LET b=INT(RND(0)*8+1)
```

```
100 CLS
110 LET ocena=5
120 FRINT "Dzień dobrv. Jak masz na imie?"
130 INFUT imie$
140 FRINT "Zobaczvmv czv znasz matematvke ";imie$;"."
150  FOR t=1 TO 5000:NEXT t
160  FOR n=1 TO 10
170   LET a=INT(RND(1)*7+3)
180   LET b=INT(RND(1)*8+1)
190   IF b>=a THEN GOTO 180
200   CLS
210   FRINT a;"-";b;"= ?"
220   INFUT c
230   FOR t=1 TO 1000:NEXT t
240   CLS
250   IF c=a-b THEN GOSUB 1000
260   IF c>a-b THEN GOSUB 2000
270   IF c<a-b THEN GOSUB 3000
280   FOR t=1 TO 5000:NEXT t
290   IF c<>a-b THEN LET ocena=ocena-0.5:GOTO 200
300  NEXT n
310 IF ocena<2 THEN LET ocena=2
320 CLS
330 FRINT "Umiesz matematvkę na";ocena;imie$;"."
340 END
1000 FRINT "                         "
1010 PRINT "                         "
1020 FRINT "                         "
1030 FRINT "    !---------------!    "
1040 FRINT "    !       A       !    "
1050 FRINT "    !       X       !    "
1060 FRINT "   /-\      X      /-\   "
1070 FRINT "  /   \     X     /   \  "
1080 FRINT " /     \    X    /     \ "
1090 FRINT "/       \   X   /       \"
1100 FRINT "!       !   X   !       !"
1110 FRINT "!";a;"-";b;"!   X   !   ";c;"   !  "
1120 FRINT "=========   X   ========="
1130 FRINT "           /X\           "
1140 FRINT "          /VVV\          "
1150 FRINT "                         "
1160 FRINT "                         "
1170 FRINT "  Bardzo dobrze ";imie$;"."
1180 RETURN
2000 FRINT "    !-_                  "
2010 FRINT "    !  -_                "
2020 FRINT "    !    -_              "
2030 FRINT "   /-\     ---_          "
2040 FRINT "  /   \     A  -_        "
2050 FRINT " /     \    X    -_      "
2060 FRINT "/       \   X      -!    "
2070 FRINT "!       !   X       !    "
2080 FRINT "!";a;"-";b;"!   X       !  "
2090 FRINT "=========   X      /-\   "
2100 FRINT "            X     /   \  "
2110 FRINT "            X    /     \ "
2120 FRINT "            X   /       \"
2130 FRINT "           /X\  !       !"
2140 FRINT "          /XXX\ !  ";c;"  !  "
2150 FRINT "                ========="
2160 FRINT "                         "
2170 PRINT "  Błąd ";imie$;"! Za dużo."
2180 RETURN
3000 FRINT "                   _-!   "
3010 FRINT "                 _-  !   "
3020 FRINT "               _-    !   "
3030 FRINT "          _---      /-\  "
3040 FRINT "        _-  A      /   \ "
3050 FRINT "      _-    X     /     \"
3060 FRINT "    !-      X    /       \"
3070 FRINT "    !       X    !       !"
3080 FRINT "    !       X    !  ";c;"  !  "
3090 FRINT "   /-\      X    ========="
3100 FRINT "  /   \     X            "
3110 FRINT " /     \    X            "
3120 FRINT "/       \   X            "
3130 FRINT "!       !  /X\           "
3140 FRINT "!";a;"-";b;"! /XXX\          "
3150 FRINT "=========                "
3160 FRINT "                         "
3170 FRINT "  Błąd ";imie$;"! Za mało."
3180 RETURN
```

POGADUSZKA

Robert Witt z Gdańska przysłał do rubryki „Sam programuję" program dydaktyczny „Pogaduszka". Robert wykorzystał elementy „Rozmówki" publikowanej w jednym z zeszłorocznych numerów „Bajtka", uzupełnił własnymi pomysłami i w efekcie powstała bardzo ciekawa i kształcąca zabawa. Program został napisany na mikrokomputerze Amstrad, myślę jednak, że użytkownicy innych komputerów nie będą mieli trudności z jego przeróbką.

Z pewnością Robert nie będzie miał nic przeciwko temu, że podzielę się z nim i innymi czytelnikami kilkoma uwagami na temat jego pracy. Zacznijmy od linii numer 40.

Instrukcja **BORDER 0** ustawia kolor ramki, natomiast **INK 0,0** kolor tła. Jeśli definiujemy te dwa parametry, warto zdefiniować również kolor napisów np. **INK 1,26:PEN 1.**

Linia 60 jest najzupełniej zbędna. Program będzie działał dokładnie tak samo gdy ją usuniemy. Warunki umieszczone w liniach 60 i 70 dopełniają się wzajemnie, a więc jeden z nich można pominąć.

W liniach 120, 125 i 126 po instrukcjach warunkowych powtarzają się te same rozkazy skoku do linii 140. Ten sam efekt można otrzymać w jednej linii programowej używając operatorów logicznych:

IF l>80 or l<2 THEN GOTO 140

Wpisałem warunek 1<2 ponieważ w przypadku podania wieku równego 1, w linii 130 komputer komunikuje „Hmm, wyglądasz najwyżej na 0 lat!", co nie jest zbyt sensowne.

Na początku programu komputer rozpoznaje czy rozmawia z dziewczyną, czy z chłopakiem, później jednak używa wyłącznie rodzaju męskiego. Spróbujcie to zmienić, jeżeli będziecie mieli trudności, wróćcie do programu „Rozmówka".

Linię 180 można zastąpić prostym skokiem

GOTO 200

gdyż skoro komputer przeszedł przez linie 160 i 170, to wiadomo, że **p$** nie jest równe ani „tak", ani „nie".

Zadaniem instrukcji warunkowych w liniach 230 i 240 jest zakwalifikowanie grającego do odpowiedniej grupy wiekowej. Jest tu pewna nieścisłość. Autor bierze pod uwagę tych, którzy mają więcej niż sześć i mniej niż dziesięć lat, oraz tych, którzy mają więcej niż 10 lat. A co z dziesięciolatkami i prawdziwymi przedszkolakami? W przypadku podania takiego wieku program przechodzi do wersji dla zakresu między 6 i 10, ale chyba nie to było celem autora.

Podprogramy dla starszych (linie 380—500) i młodszych (linie 250—370) praktycznie nie różnią się niczym, prócz parametrów pętli. Nie jest więc potrzebne umieszczanie ich w programie dwukrotnie. Proponuję następujące rozwiązanie:

230 IF l<7 then goto 240
235 IF l<10 then pocz=1:kon=10:goto 250
237 pocz=10:kon=20
240 PRINT "Jesteś za mały"; o$:END
250 FOR n=pocz TO kon

Po wprowadzeniu tej zmiany możemy usunąć wszystkie linie od numeru 380 w górę.

* * *

Myślę, że moje uwagi przydadzą się nie tylko Robertowi ale i wam wszystkim. Robertowi zaś gratuluję i życzę wielu ciekawych pomysłów. Tak trzymać!

Romek

```
10 REM Pogaduszka
20 REM Robert Witt
30 CLS
40 BORDER 0:INK 0,0
50 INPUT "Jak masz na imie":o$
60 IF RIGHT$(o$,1)="a" THEN 80
70 IF RIGHT$(o$,1)<>"a" THEN 90
80 PRINT "Chyba jesteś fajna dziewczyna,
 ":o$:"!!!":GOTO 100
90 PRINT "Chyba jesteś fajnym chłopcem.
":o$:"!!!":GOTO 100
100 PRINT "A ile masz lat"
110 INPUT l
120 IF l>80 THEN GOTO 140
125 IF l=0 THEN GOTO 140
126 IF l<0 THEN GOTO 140
130 PRINT "Hmm, wyglądasz najwyżej na";I
NT(l*0.9);"lat!":GOTO 150
140 PRINT "Nie wygłupiaj sie!!:":GOTO 11
0
150 INPUT "Czy chciałbyś się ze mna uczy
ć ";p$
160 IF p$="tak" THEN 210
170 IF p$="nie" THEN 190
180 IF p$<>"tak" AND p$<>"nie" THEN 200
190 PRINT "Jesteś bardzo leniwy!":END
200 PRINT "Nie wiem co to znaczy!":GOTO
150
210 PRINT "Podaj liczbe do mnożenia"
220 INPUT i
230 IF l>6 AND l<10 THEN GOTO 250
240 IF l>10 THEN GOTO 380
250 FOR n=1 TO 10
260 PRINT n"*"i"=";
270 INPUT w
280 IF w=n*i THEN 350
290 PRINT "źle! Jeszcze raz!"
300 INPUT w
310 IF w=n*i THEN 330
320 PRINT "Jednak z matmy jesteś słaby."
:END
330 PRINT "No, teraz już dobrze."
340 GOTO 360
350 PRINT "Bardzo dobrze."
360 NEXT n
370 PRINT "Dałeś rade!":END
380 FOR n=10 TO 20
390 PRINT n"*"i"=":
400 INPUT w
410 IF w=n*i THEN 480
420 PRINT "źle, jeszcze raz!"
430 INPUT w
440 IF w=n*i THEN 460
450 PRINT "Jednak z matmy jesteś słaby."
:END
460 PRINT "No, teraz już dobrze."
470 GOTO 490
480 PRINT "Wyśmienicie!"
490 NEXT n
500 PRINT "Dobry jesteś z tabliczki mnoż
enia.":END
```


CO PAN NA TO, PANIE BELL?

Dokończenie ze str. 32

skiem spodni i co umożliwia zdalne odsłuchanie nagranych rozmów. Łącząc się z dowolnego aparatu ze swoim numerem, po usłyszeniu własnej wstępnej zapowiedzi, do mikrofonu przytyka się „beeper". Jego ostry dźwięk przekazany łączami telefonicznymi uruchamia odczyt. Jeśli raz jeszcze chcemy przesłuchać zostawione nam wiadomości, „beeper" zrobi swoje po raz kolejny.

Następnym krokiem naprzód jest system „pager". Za przydzielenie dodatkowego numeru „pagera" płaci się dodatkowy miesięczny abonament. Ale dla ludzi interesu, których czas liczy się sporymi kwotami ma to kolosalne znaczenie. Na ciężarówkach poważnych firm, biurach adwokackich, gabinetach lekarskich za Atlantykiem wymalowane są, prócz zestawu telefonicznych cyfr, numery „pagerów".

Kontakt z lokalną centralą systemu „pager" i wywołanie numeru abonenta uruchamia brzęczyk jego „beepera", który każdy zapobiegliwy biznesmen trzyma przy sobie w każdej sytuacji. Na plaży, w saunie, podczas gry w golfa. Sygnał oznacza, że ktoś nagle próbuje się z nim porozumieć. Od jego woli zależy już tylko, jak szybko dotrze do najbliższego aparatu, połączy z własnym domem, lub centralą „pagerów" i dotrze na linii do poszukującej go osoby. Od dyżurnego systemu może zażądać treści informacji, albo otrzymać numer, z którym należy się skontaktować. Już teraz pracuje się nad nowymi udogodnieniami. W przyszłości zmodyfikowany „beeper" będzie nie tylko alarmował, ale także od razu podawał dane aparatu poszukującego, a nawet przekazaną informację.

Tyle z autopsji. Amerykański tygodnik „Newsweek" doniósł niedawno, iż w Stanach Zjednoczonych czyni się starania aby wyeliminować zasadniczą wadę systemu „pager" — niewielki zasięg całego układu. Nadajniki radiowe transmitujące sygnał alarmu dla posiadacza „beepera", łapią go tylko wówczas, kiedy znajduje się nie dalej niż 20—30 mil (32—48 km) od centrali. A cóż będzie, gdy nasz człowiek interesu wyjedzie poza własne miasto?

Właśnie o tym pomyślała firma National Satellite Communications z Cleveland w stanie Ohio. Do pracy zaprzęgnięto, jak się łatwo domyślić pojazdy kosmicznej łączności zawieszone nad USA. Za sumę od 45 do 70 dolarów amerykańskich miesięcznie klient otrzymuje bardzo, bardzo wymyślny „beeper" — czarne pudełeczko o masie dwóch uncji. Zabiera je ze sobą w każdą podróż po Stanach Zjednoczonych i bliskiej zagranicy.

Ktoś chcący naradzić się z ruchliwym biznesmenem łączy się... za darmo z numerem sztabu komputerowego systemu w Waszyngtonie, wybiera następnie klawiszami własnego aparatu kod delikwenta i przekazuje wiadomość. Resztą zajmuje się elektronika. Sygnał trafia do satelity, a dalej do sieci małych emiterów naziemnych pokrywających swym zasięgiem całe USA. Na 200 kanałach „beeper" automatycznie wyszuka najsilniejszy sygnał. Częstotliwości są tak dobrane, iż czarna skrzyneczka odbierze i pokaże swemu właścicielowi numer tego, kto pragnie się porozumieć (może przyjąć jednocześnie do 6 różnych zgłoszeń z różnych rejonów kraju). To usługi podstawowe tego, co w całości nosi nazwę Cue.

Za dodatkową opłatą abonent Cue nie potrzebuje się łączyć z tym, który wysłał telefoniczny „news". Dzwoni jedynie do centrum w Waszyngtonie, a komputer modulowanym głosem przekazuje mu oczekiwaną informację. I ten rachunek telefoniczny wędruje na konto Cue. Jedynie rozmowy zagraniczne obciążają dzwoniącego.

Ale to nie koniec. Już teraz, gdy Cue dopiero się rodzi pomyślano o dwóch kierunkach rozwoju. Dwóch celach przyszłościowych. Pierwszy to przekazywanie od razu „beeperowi" głosu z telefonu poszukującego kontaktu. A drugi to rozszerzenie usług na cały... glob.

Wojciech Łuczak

NIE TYLKO KOMPUTERY

Z niezwykle prostym wynalazkiem Grahama Bella zrobiono już prawie wszystko. Zaczęło się wszak od dwóch tub połączonych przewodami. Dopiero później pojawiły się łącznice i panienki telefonistki. Właśnie wówczas po raz pierwszy użyto sformułowania, iż czarodziejska skrzyneczka z korbką najbardziej usatysfakcjonowała piękniejszą połowę społeczeństwa. Ale kiedy na scenę wkroczyły automatyczne centrale, podmorskie kable, zaprzestano dowcipów. Telefon stał się podstawowym narzędziem robienia interesów.



Dziś zwoje miedzi ułożone wewnątrz grubego, niczym udo mężczyzny, kabla, pieczołowicie umieszczone kilka kilometrów pod lustrem oceanu to już jedynie przeszłość techniki. Dziś miedź ustępuje pola światłowodom, a słowa wypowiedziane gdzieś w londyńskiej budce telefonicznej, nim trafią powiedzmy do Tokio, czy Los Angeles, dotrą do satelity zawieszonego na orbicie i w mgnieniu oka pokonają w kosmosie dystans, który poprzednio przebywać musiały po dnie oceanu. Dziś większość cywilizowanego świata uważa posiadanie sprawnego systemu umożliwiającego błyskawiczny kontakt praktycznie z każdym punktem kuli ziemskiej, za coś tak normalnego, jak widok mydła w łazience. Niestety nie należymy do tej większości, przyjrzyjmy się więc, jak w rękach innych działa coraz doskonalszy telefon. Bo to przecież wielce pouczające.

Zacznijmy od prostego mariażu mikroprocesora z nieco tylko uwspółcześnionym wynalazkiem Bella. Taki właśnie aparat, choć trefnisie mogliby go określić jako „szkocką machinę", zobaczyłem po raz pierwszy na początku lat siedemdziesiątych w Środkowej Anglii. Kiedy nikt się nim nie posługiwał, podawał wyłącznie czas. Każde połączenie wywoływało jednak reakcję jego „magicznego oka" z błyskającymi cyframi.

— Za półtora funta zapytamy o pogodę w Kalifornii — rzekł mój brytyjski przyjaciel. Po sześciu cyfrach miły dziewczęcy głos z taśmy — „Jesteś na linii satelitarnej". Dalsza kombinacja tarczą i mamy wreszcie naszego znajomego znad Pacyfiku. Zamieniamy tylko kilka zdań, bo dość szybko w zielonym okienku pojawia się odczyt 1,5. Zegar przemienił się w licznik pieniędzy z własną inteligencją. W ten sposób można chyba zamknąć usta najbardziej gadatliwym.

Upłynęło trochę czasu nim w mieszkaniach tych najbardziej podróżujących Polaków znalazły się najnowsze telefoniczne cudeńka. Klawiszowe aparaty z rozbudowaną pamięcią nikogo już nie szokują, via Singapur i Hongkong zjawiają się na bazarach i komisach. Dość rzadkie są telefony „z antenką" — łączące nawet w odległości kilkuset metrów od zainstalowanej i połączonej z siecią ogólną minicentralki. Ale prywatne aparaty odbierające wezwania podczas nieobecności gospodarza i nagrywające wiadomości dla niego są u nas dostępne naprawdę nielicznym.

Te podstawowe „odpowiadacze" są na kontynencie amerykańskim bardzo, bardzo popularne. Tym bardziej, iż ich masowa produkcja (firmy ze Stanów Zjednoczonych, Kanady plus zalew sprzętu montowanego w Japonii, krajach ASEAN-u) pozwoliła poważnie zredukować ich cenę. W Kanadzie w sklepie sieci Radio Shack podstawowy model „odbieracza telefonów" kosztuje około 200 tamtejszych dolarów. Jego serce tworzą dwa magnetofony kasetowe. Na pierwszym nagrywa się tekst audiowizytówki, w rodzaju: „Cześć, nie ma mnie w domu... po usłyszeniu sygnału zostaw swój numer telefonu i wiadomość" (regulowana jest liczba dzwonków, po której do akcji wchodzi automat). Drugi rejestruje zgłoszenia. Po dłuższej, czy krótszej nieobecności, w rodzinach kanadyjskich, które obserwowałem, pierwsze kroki kierowane są właśnie w stronę urządzenia zapisującego telefony.

Właściciele małych, kilkuosobowych przedsiębiorstw reklamujących się w codziennych gazetach, bądź zarejestrowani w profesjonalnych książkach telefonicznych usługodawców — „yellow pages" nie wyobrażają sobie pracy bez rejestratora telefonicznych zgłoszeń. Właśnie dla nich wymyślono bardziej doskonałą odmianę „odpowiadaczy" droższą o około sto dolarów od wariantu „Standard".

Za tę sumę prócz pudełeczka z magnetofonami połączonego z telefonem kupuje się jeszcze „beeper", czyli coś, co nosi się zatknięte za pa-

Dokończenie na str. 31

CO PAN NA TO, PANIE BELL?